Robust approaches for scheduling under uncertainties: the RobOCoop project and its relevance for grid computing

Eric sanlaville

eric.sanlaville@univ-lehavre.fr

june 2009

Plan

Plan

Plan

Plan

Plan

1. Scheduling under Uncertainties : application domains, solving methods
2. Robocoop project
3. Cooperative and robust scheduling for the grid
4. Looking for dynamic and distributed methods for the grid

Plan

Uncertainty sources

- Approximation
  - durations (eg, task durations, communication delays); costs; resource characteristics (speed)
- Disturbance
  - resource breakdown
  - task arrival or removal
- Ignorance (foundamental)
  - natural phenomena, behaviour of other actors : distributed decision process !!

Uncertainty sources

- Approximation
  - durations (eg, task durations, communication delays); costs; resource characteristics (speed)
- Disturbance
  - resource breakdown
  - task arrival or removal
- Ignorance (foundamental)
  - natural phenomena, behaviour of other actors : distributed decision process !!

Application Domains

- Industry : workshops, supply chain
- project management, planning (building, marketing, timetabling,...)
- parallel computing

Uncertainty Models

- probabilities : some parameters are random variables (independent?)
- scenarios : some parameters have a set of possible values, no probability attached
  - finite set
  - interval set

  A scenario is obtained by fixing each parameter (independently or not). Sometimes a probability is attached to ONE scenario.
- fuzzyness : some parameters are fuzzy sets.

Scheduling history : early attempts by probabilistic scheduling
Recent works : robust optimization, stochastic programming.

Solving Approaches

How to classify ?

- Time of computation
  At what time is the solution computed? before, during execution?
- Uncertainty model
  see above
- Performance evaluation
  linked to the model, yes. What else?

Proactive versus reactive approaches

- Proactive : a solution is computed a priori. But the algorithm takes into account the uncertainty.
- Reactive : the final solution is built at execution time (during execution of the schedule); the decisions are taken as late as possible !
- Proactive Reactive approach : The solution is partially built before execution time. The set of decisions is completed (modified?) later.

- Not specific to scheduling (eg, two phase stochastic programming). But scheduling deals with times!
- In many applications, a reactive part is mandatory. True for scheduling with disturbances.
- One may speak of building process instead of algorithm.

Proactive versus reactive approaches

- Proactive : a solution is computed a priori. But the algorithm takes into account the uncertainty.
- Reactive : the final solution is built at execution time (during execution of the schedule); the decisions are taken as late as possible !
- Proactive Reactive approach : The solution is partially built before execution time. The set of decisions is completed (modified?) later.

- Not specific to scheduling (eg, two phase stochastic programming). But scheduling deals with times!
- In many applications, a reactive part is mandatory. True for scheduling with disturbances.
- One may speak of building process instead of algorithm.

Proactive versus reactive approaches

- Proactive : a solution is computed a priori. But the algorithm takes into account the uncertainty.
- Reactive : the final solution is built at execution time (during execution of the schedule); the decisions are taken as late as possible !
- Proactive Reactive approach : The solution is partially built before execution time. The set of decisions is completed (modified?) later.

- Not specific to scheduling (eg, two phase stochastic programming). But scheduling deals with times!
- In many applications, a reactive part is mandatory. True for scheduling with disturbances.
- One may speak of building process instead of algorithm.

Performance evaluation : from the classical criteria...

- makespan : on the set of tasks
- flow time : individual. measures the presence time of each task
- stretch : the same, but weighted by the processing time. Measures the response time of the system.

These criteria may be computed on several instances. Their values may be compared with absolute thresholds, with optimal values computed a posteriori,...

mean, max, deviation (absolute or relative).

example of maximum absolute deviation : $\max_{\mathcal{I}} \frac{Z(\mathcal{A})-Z^*}{Z^*}$

- There is a very large number of performance indicators !

Performance evaluation : ... to global criteria considering the whole schedule building process

- Nervousness : decisions may be changed during the process. Too often?
- Response time of the building process: respect of time scale of application. Grid Scheduling !!

How to evaluate a schedule building process in presence of uncertainties is not trivial !!

Plan

Aims

ROBOCoop : Robustness of Cooperative Schedules

- Better understand and classify the different approach for robust scheduling
- Look for robust AND cooperative approaches for different scheduling problems
- Build a tool for evaluating approaches in presence of uncertainty.

resources

ROBOCoop is a 3 year project funded by the french ANR agency. Partners:

1. LI Tours (OC) : shop scheduling
2. LAAS Toulouse (MOGISA) : planning, cooperative scheduling (1 PhD student)
3. LITIS Le Havre (RI2C, and LIMOS): parallel computing, Decision Aid, complex dynamic systems (1 engineer)
4. ILOG (ILOG scheduler) : constraint programming, case studies.

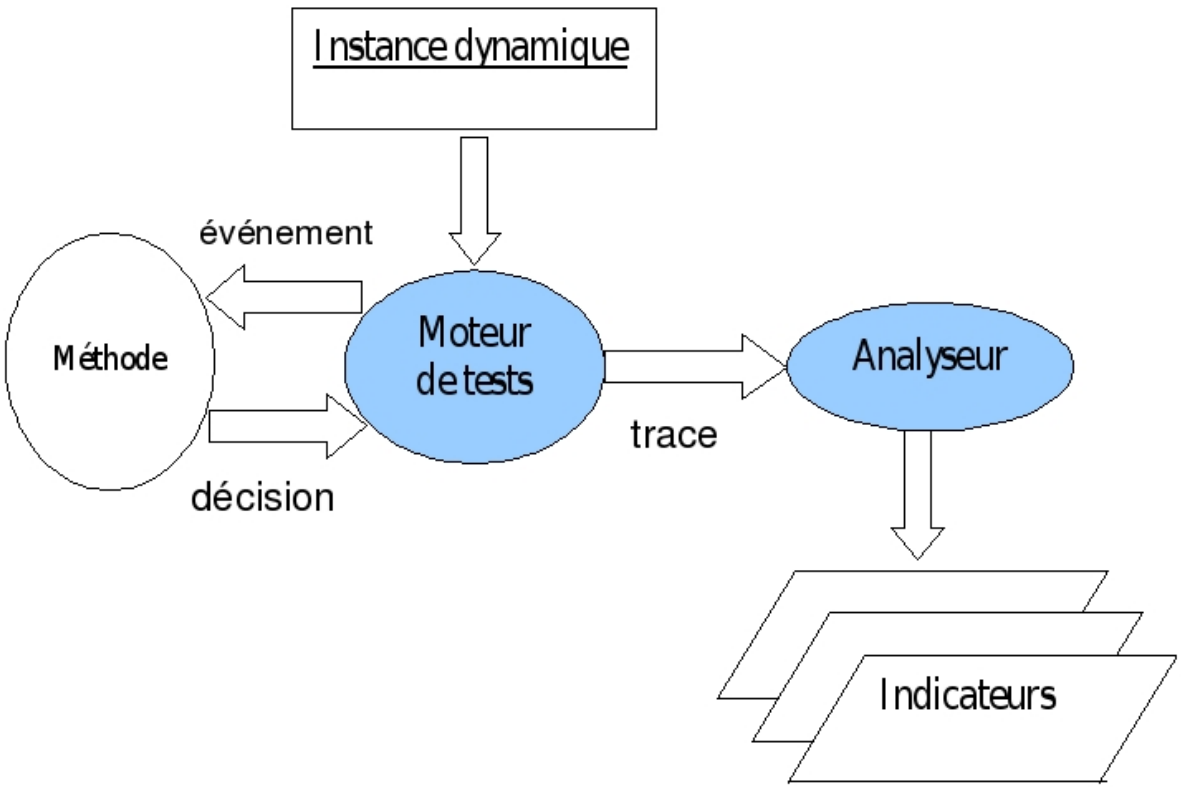
History :

- since 1999, a group of a dozen of french (plus one belgian) teams on Flexible and Robust Scheduling.
- 2003 : A project of CNRS specific action on OR.
- 2008 : Flexibility and Robustness in Scheduling, ISTE-Wiley.

Why robust cooperative framework ?

- Complex planification problems : several decision centers
- Distribution of decisions involves uncertainties at each decision center.
- Belief : the best results will be obtained through cooperation mechanisms.

Software Structure of the evaluation tool



Design of dynamic instances

The user specifies the problem its method solves (eg, m machines, precedence constraints, communication times), and the uncertainty model (eg, communications modelled by time intervals, length at most 10% of the minimum value).

- Dynamic instance : a set of couples (t, e)
  - t : a date (in the execution time scale
  - e : an event (task arrival, duration modification, resource unavailability notification, etc...)
- Choice : start from a static instance (eg, PSP-LIB). Disturb it !

Test Engine

It dialogues with the tested method

- From the dynamic instance generator, provides to the method the event e at the specified time t.
- Receives from the method the decisions taken
- Builds the log, and passes it to the trace analyst.

Performance indicators

The goal : to provide the user with a large battery of indicators

- global ones : response times, nervousness, flexibility...
- if possible : deviation (or estimate) with regard to the criteria wished by the user (eg, makespan and flexibility)
  - the dynamic instance can be kept with previous or best result.
- If repeated tests : mean values, worst case studies,...

Plan

Specifics of the grid

The example grid is EGEE (Europe, now world wide) : Enabling Grids for E-Science (start 2001).
Initially designed to provide computation and storage resource for exploitating data from LHC experiments.

- 140 institutions, 300 sites (nodes), 50 countries
  Node: a cluster of CPU (PE's), some storage units (SE's). Some clusters group more that 1000 CPU's, some 20.
- 130 VOs (Virtual Organizations):
  a group of users, geographical or thematic. The members of one VO have access to a subset of the grid nodes. ex : Auvergrid, Atlas, Biomed
- 90 000 CPUs
- 120000 TB

Egee grid : nowadays management

Centralized management : General Resource Broker (CERN). One Resource Broker by VO. All informations about the site: state, load, is centralized.

- Problem 1 : a very large amount of demands and data to the RB.
- Problem 2 : keeping information up-to-date? (eg, the RB sends new jobs to a site that just "collapsed")
- Problem 3 : dynamic balancing of the load is very heavy

Towards distributive management ?

From an "external" (and internal!) point of view : a distributed management would avoid these difficulties

- First job assignment : left to resource brokers?
- Load-Balancing : via a collaboration between site managers?
- local scheduling : still decided locally.

$\Rightarrow$ It is desirable to test distributed, collaborative tools, at least for load balancing.

Some possible approaches for the dynamic load balancing

- dynamic distributed load balancing: the classical examples from parallel systems
- highly dynamic centralized load balancing
- proactive centralized approach
- towards a proactive/reactive centralized/distributed approach?

Plan

Dynamic distributed load balancing

- Point of view : real time management of a parallel system
  - parallel system : a set of processes (tasks), a set of heterogeous machines, a network
  - processes are created on the different machines
- Goal : propose a migration policy to keep the load balance

Dynamic distributed load balancing: one example of model

- process: unit size, amount of memory use. no precedence
- machine; speed, memory capacity
- network: cost of transfer from M_i to M'_i
  network constraint: minimize the overall cost of transfer?
- load measurement per machine: number of present processes divided by speed.
  Note : can be complicated by other factors, as the use of the machine by an interactive user.

Sources: Perotin 2008, Aggarval et al 2003, Xu and Lau 1997, Kunz 1991,...

The local scheduler

At time t, it decides of the processes to execute locally or to migrate.

- periodically awaken
- local knowledge: its load. can exchange with other machine schedulers.
- local algorithm : computes which process to migrate to which machine IT SOLVES THE PROBLEM AT TIME t !!
  1. awakes at time t
  2. computes its load at t
  3. sends its load value to all other machines
  4. computes what to do
  5. sends the processes
  6. sleeps until time $t + \delta$

example of local algorithms

To avoid bicriteria, a constraint is added on the maximum cost of transfer during one phase B. Hence B represents the largest set of processes that can be transferred during δ time units.

- simple algorithm
  1. If I am the most loaded then
  2. send my smallest process to the least loaded machine M_i
- small variant :
  If M_i can not receive p_j then sends p_j to another possible machine.

example of local algorithms

- Rudolf inspired algorithm
  1. If I am the most loaded then
  2. computes M_i, the machine that can accept the largest number of processes from me and stay with a number of processes less than or equal to me
  3. migrates the processes.

Sources: Perotin 2008, Rudolf et al 1991

example of local algorithms

- relaxation based algorithm RBT
  If there is no memory constraint on the machines, and no network constraint, the problem is easy: locate *n* tasks to *m* uniform machines. The "ideal" load can be rapidly computed.
  1. If I am the most loaded machine then
  2. compute for each machine its ideal load.
  3. transfer the smallest process to a less loaded machine that can accept it. Do it until I am at ideal load, or the network constraint is reached.

Sources: Perotin 2008

Comparing the local algorithms

- Speed: Rudolf and RBT are much slower. Hence : constant δ larger !
- Efficiency (Perotin 2008 with SimGrid) global measures: the mean load, the number of processes finished,...
  - simple algo: much better than nothing (no transfer)
  - Rudolf : very efficient to balance the memory load. Accepts more jobs (bounded queues)
    Side effect : because of preemption (processor sharing) with equal priority, Jobs finish later !
- All algos send processes from the most loaded machine. In some cases, balance is impossible with such moves only !!
- All algorithms are myopic : they make instantaneous optimization. What about the global performance?

Conclusion for local algorithms

- It works reasonably well for independent, small tasks of same size.
- Easy to implement on local networks.
- No performance guaranty
- No possible use of application structure, duration distributions.
- What for grids: networks of clusters? to migrate from one cluster to another?

A cyclic scheduling model

This model considers the mangement of one cluster only. It is adapted to some kinds of VO's: data from the LHC.

- user : member of a VO, sends bags of jobs
- job: periodic release date, estimated duration (precedence?)
- machines : identical
- first goal: feasibility
- performance : minimize the presence time of each user. Measure the equity ?!?

This approach might be considered as proactive (the task durations can not be accurately predicted). It needs to be coupled with some collaborative mechanism with the other sites.

Sources: Medernach 2009

An example of more exotic approaches

Optimization by Ant colony

Source : Antoine Dutot PhD Thesis, Le Havre, 2006.

Algorithms based on numerical ants

Application to organization detection in systems made of a large number of interacting entities.

And use this organization detection ability to :

- Distribute dynamically and adaptively an application
- But also to do community detection
- And more ...

Criterions for the distribution

There is a constant struggle between :

- load unbalance;
- communications.

→ We must find a tradeoff.

However, in an application made of multiple interacting entities, these entities:

- tend to communicate more with some than with others;
- organize themselves, auto-organize themselves, and form groups.

→ organizations appear.

Organizations

Main idea: Use these organizations to better distribute the application, that is minimize communication costs.

Organizations may last whereas, their constituents appear, evolve and disappear.

$\rightarrow$ However we must also balance the load, that is, distribute organizations the better we can on computing resources.

$ANTCO^2$: Principle

Collaboration and competition

Main idea: use collaboration and competition mechanisms.

1. As many colonies as computing resources;
2. each colony owns an unique color;
3. iterative algorithm, ants move inside the graph from one vertex to the other, crossing one edge at each iteration.
4. they lay down color pheromones of their colony's color on each crossed edge.

Competition is used to balance the load: colonies compete to take parts of the graph by coloring the graph. Equilibrium should balance the number of vertices in each part.

Collaboration is used to minimize communications: Inside one colony, ants cooperate to detect organizations, using stigmergy.

Move

Ants move at each time step according to probabilities.

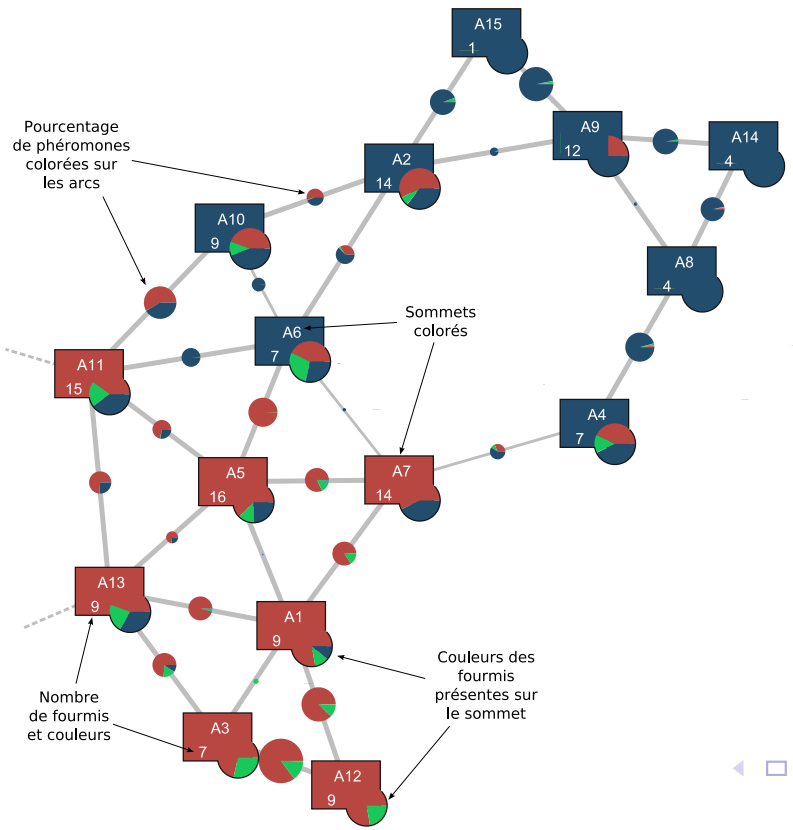
Such moves allow to detect organizations by pheromone dropping. Indeed :

- Ants are attracted by edges with high weights.
- Ants are attracted by pheromones of their own color.
- Ants are repulsed by other colors.
- Organizations are ”marked“ with pheromone.
- This way of doing, using probabilities, allows fluctuations, and thus adaptivity.

Coloration

Colonies colonize the graph by coloration, using their pheromones.

We establish the color of a vertex by computing the dominant pheromone on all adjacent edges.



Objective function ?

→ There is no objective function helping to build the solution.

- ants do not iteratively build paths that will be evaluated in a final step.
- → they collectively maintain homogeneous colored zones.
- They only make local choices.
- Le problem evolves dynamically.
- The solution is not built, but maintained inside the graph, in an adaptive fashion taking continual reconfigurations into account. Indeed, the solution is a *side effect*.

Distributed version of *ANTCO*^{2}

- One version of *ANTCO*^{2} at each node.
- Local subgraph: the vertices associated to tasks executed by the node, plus "virtual nodes" representing adjacent subgraphs.
- Start : an ant colony on each node.
- If an ant moves to a virtual node, it is "physically" moved to the corresponding node. If a vertex is colonized by a color not associated to its node :
  A transfer decision can be done.