Simulation approaches for optimization in business and service systems

Imed Kacem

kacem@univ-metz.fr

Professor - Université Paul Verlaine – Metz

http://kacem.imed.perso.neuf.fr/site/

FUBUTEC 2011, Future Business Technology, April 18-20, 2011, British Institute of Technology and Ecommerce, London, UK

eurosis

Outline

- Some problems in business and service systems
- Common structure
- Complexity and diversity
- Used approaches (exact and heuristic)
- Example
- Conclusions

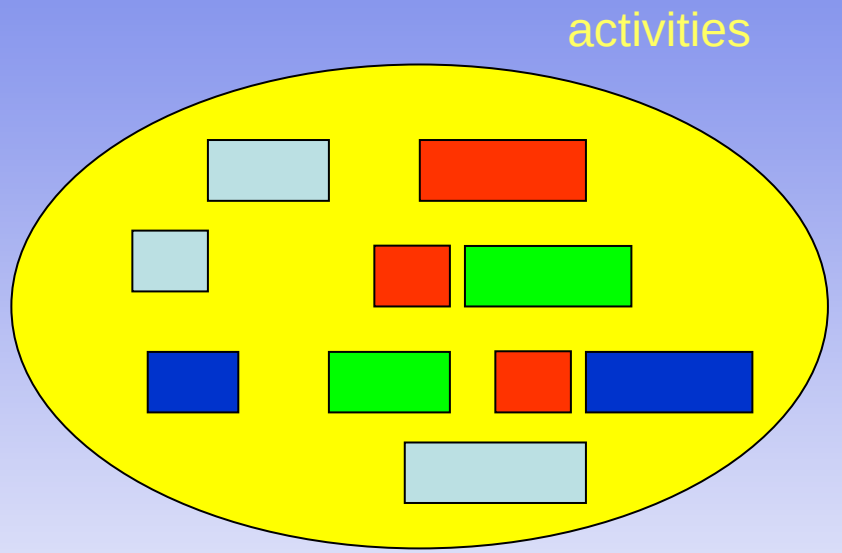
Some problems in Business and Service Systems

Portfolio selection

We have a set of activities to carry out. Each of them can generate a known profit. Our production system has a limited capacity and it cannot handle all the activities.

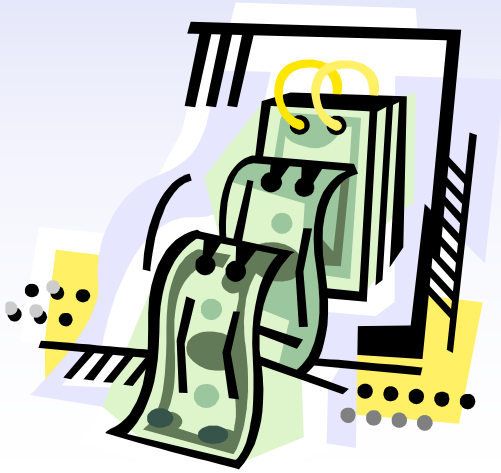
The question is how can select the good subset of activities by respecting the production system contraints, in order to maximize the total profit.

Similar problem: efficient diversification of investments.

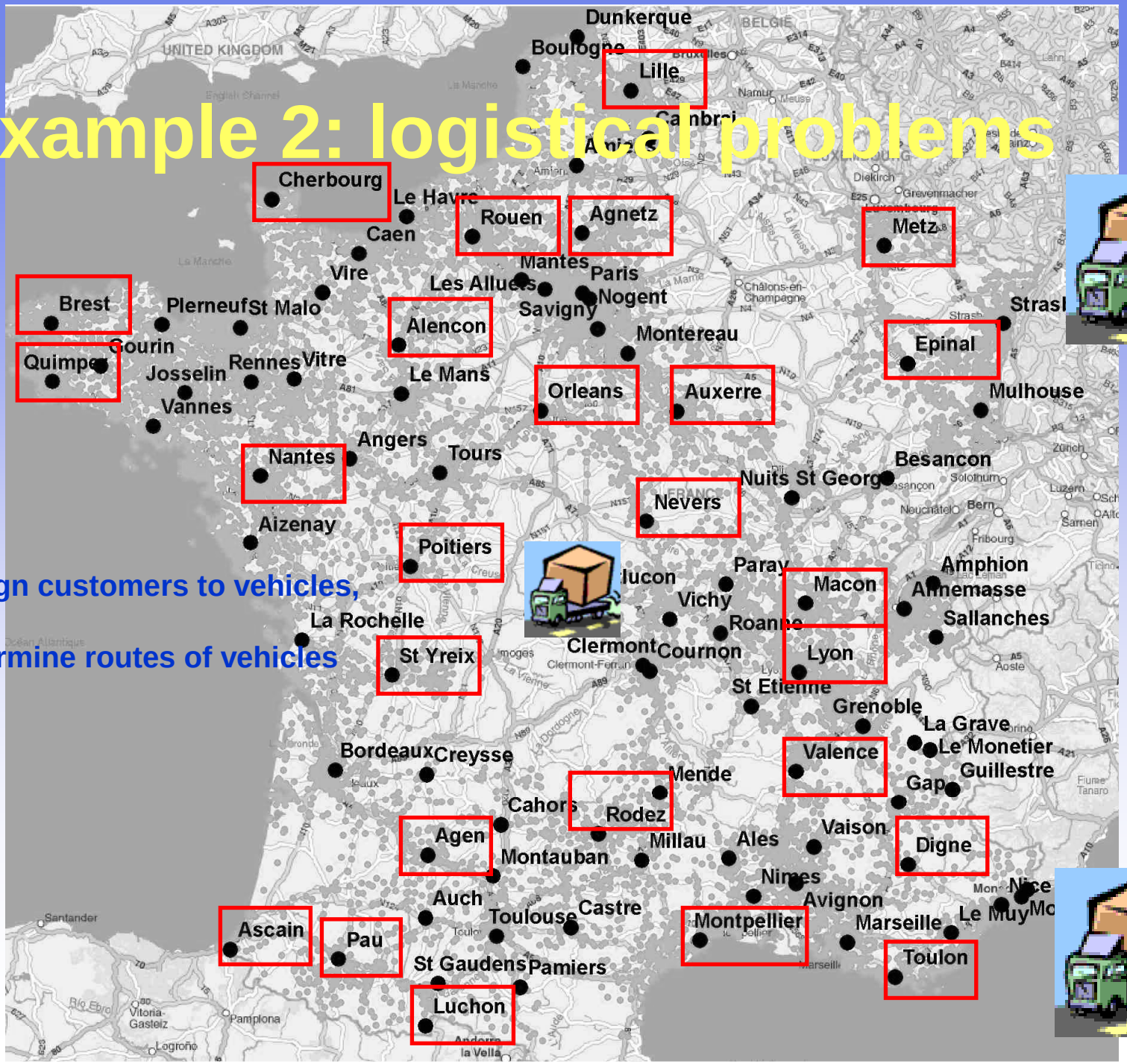


Production system

with a limited capacity

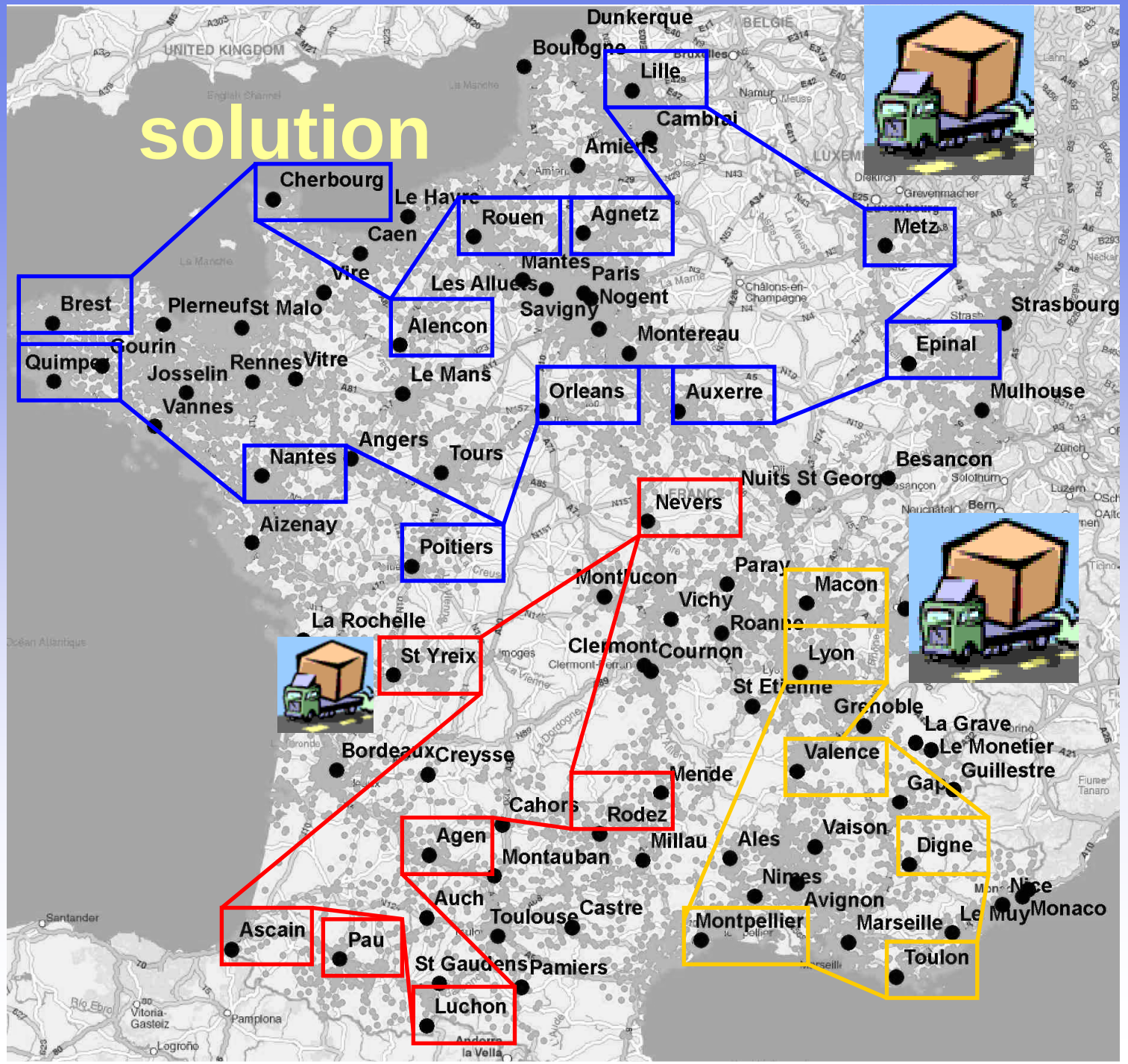


Example 2: logistical problems



Assign customers to vehicles,

Determine routes of vehicles

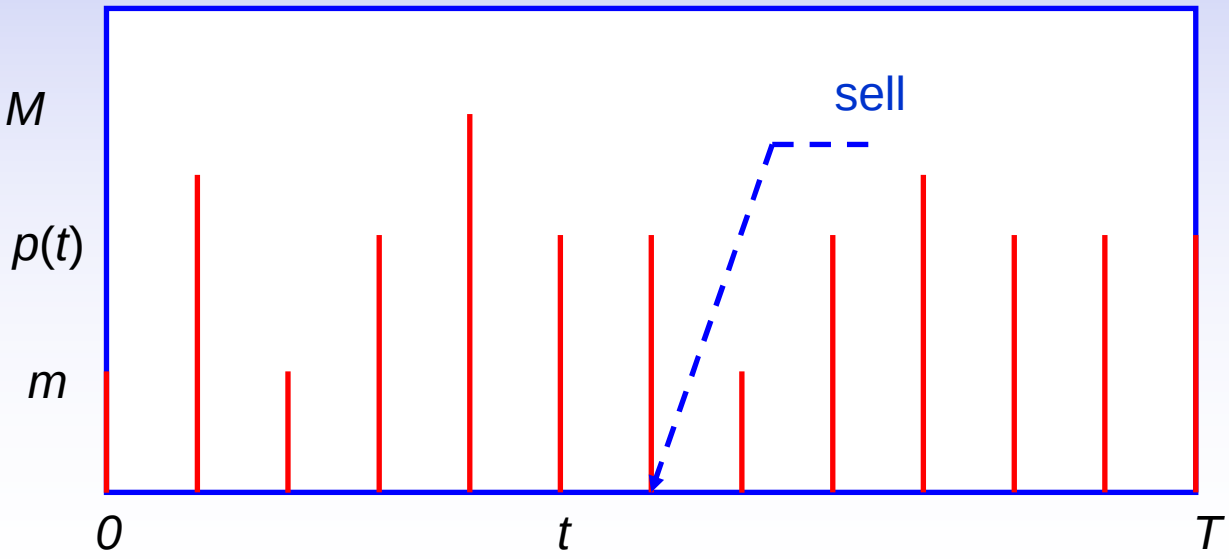

solution
Dunkerque
Boulogne
Lille
Cambrai
Amiens
Cherbourg
Le Havre
Caen
Rouen
Agnetz
Metz
Vire
Mantes
Paris
Les Alluets
Nogent
Savigny
Brest
Plerneuf
St Malo
Alencon
Montereau
Strasbourg
Epinal
Gourin
Quimper
Josselin
Rennes
Vitre
Le Mans
Orleans
Auxerre
Mulhouse
Vannes
Angers
Tours
Besancon
Nuits St Georg
Nantes
Nevers
Aizenay
Poitiers
Paray
Montlucon
Macon
Vichy
La Rochelle
Roanne
St Yreix
Clermont
Cournon
Lyon
St Etienne
Grenoble
La Grave
Le Monetier
Bordeaux
Creysse
Valence
Guillestre
Mende
Gap
Cahors
Rodez
Vaison
Agen
Millau
Ales
Digne
Montauban
Nimes
Nice
Auch
Avignon
Monaco
Toulouse
Castre
Montpellier
Marseille
Le Muy
Ascain
Pau
Toulon
St Gaudens
Pamiers
Luchon

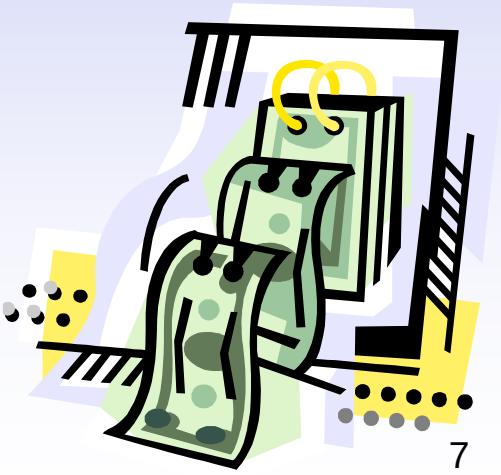
Online Trading Problems in Financial Markets

We aim to buy electronic market places at low and to sell them at high prices (see *Schmidt*, EJOR).

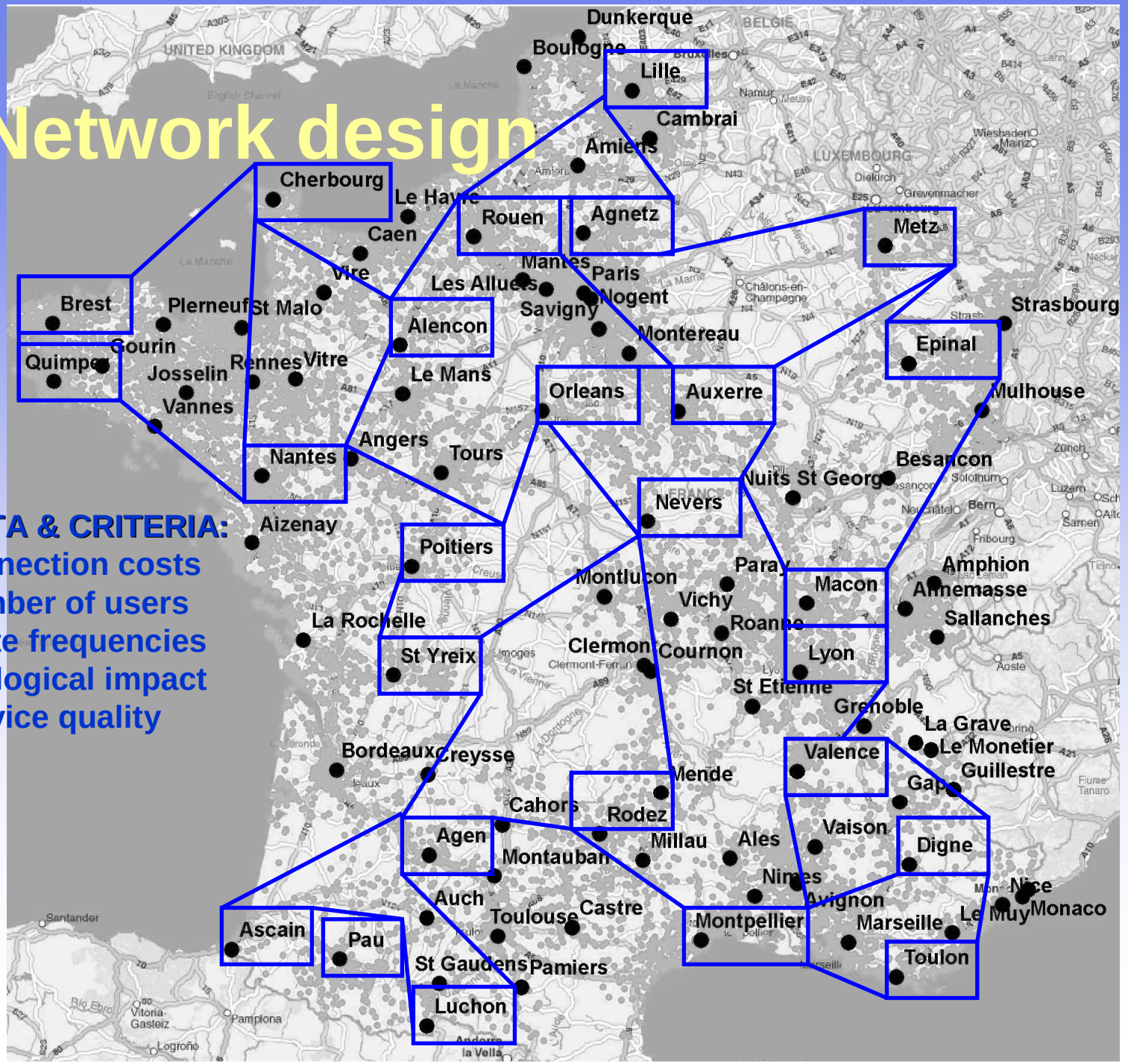
We (the trader) own an initial asset A at time $t = 0$. We can obtain dynamically a price quotation $m \le p(t) \le M$ at every time $t = 1, 2, \ldots, T$. Parameters m and M are known in advance. Hence, we have to decide at time t if accept this price for selling. Trading is closed once we accepted some $p(t)$. If we did not accept any price until time $T - 1$ we will be obliged to accept the last proposed price at time T .

General problem: buy and sell at given periods.





Network design

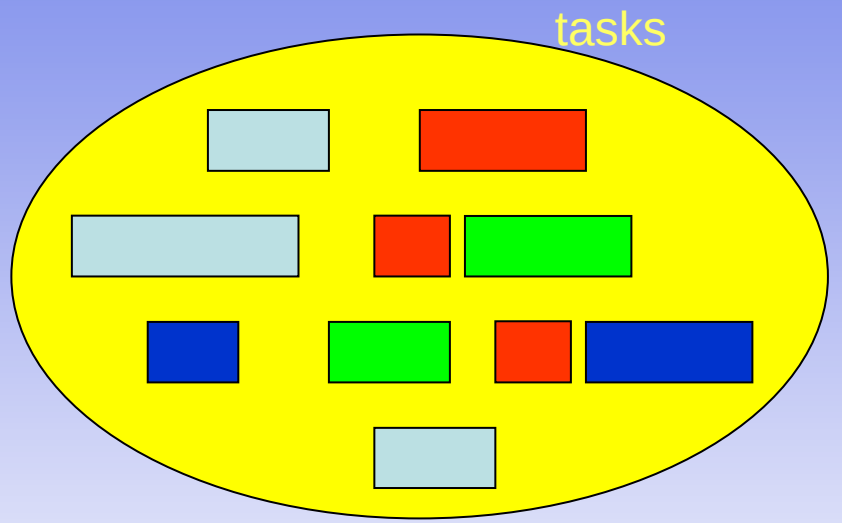


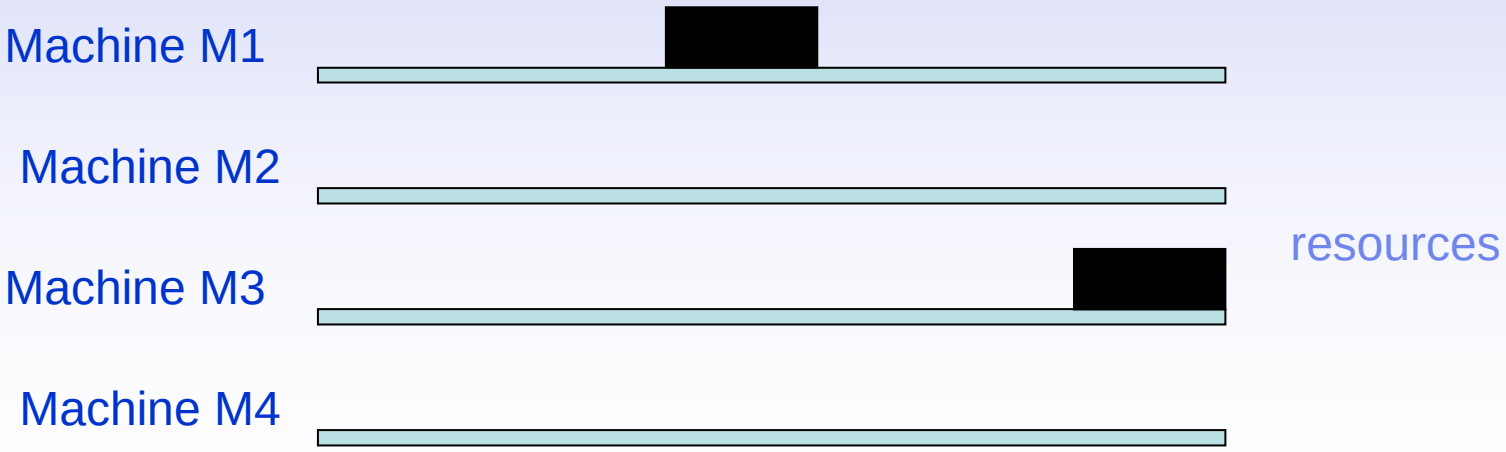
DATA & CRITERIA:

connection costs
number of users
route frequencies
ecological impact
service quality

Scheduling problems

solving a scheduling problem can be reduced to the organization of a set of activities (jobs or tasks) by exploiting the available capacities (resources). This execution has to respect different technical rules (constraints) and to provide the maximum of effectiveness (according to a set of criteria or objectives). (see *Carlier & Chrétienne*)





Example: the organization of a transportation network

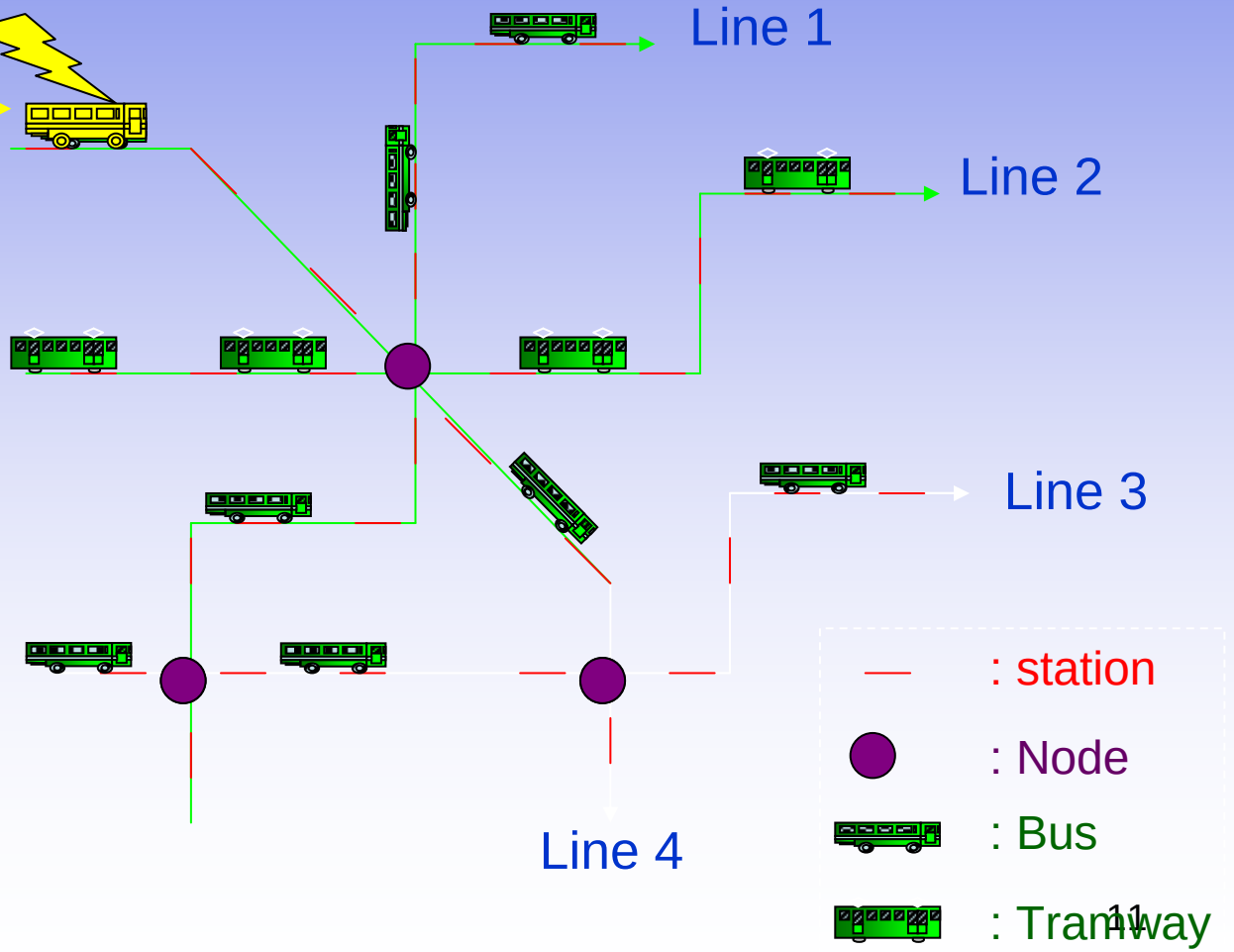
- ***Here, the problem consists in establishing a planning indicating the different schedules of buses and stations to be served.***
- ***The buses, which represent the resources, have to pass at precise times, by precise stations according to a fixed order (precedence constraints).***
- ***The tasks consist hence in transporting the travellers from a station to another.***
- ***At every moment of the day, the number of travellers waiting in a station is variable. The frequency and the capacity of buses assigned to the used ways must be coherent with the objective. One can find other types of constraints as those imposed in the case of a correspondence (in this case, the arriving bus has to wait for another bus so that some travellers will be able to reach their destination).***

Example: the organization of a transportation network *(Dridi & Kacem 2004)*

- organize the transportation network

Objective

- schedule the trips in order to:
  - Maximize the service quality
  - Respect the temporal contraints



Common structure

Discrete Optimization Problems

- ***The mentioned problems has a common structure. This structure is based on the four following elements:***

1. ***Parameters:*** ***The parameters are all the necessary data to define the problem and the constraints.***
2. ***Decision variables involving discrete variables (binary variables and integer variables) :*** ***They represent the decisions to take in order to construct a feasible solution and to optimize one or several criteria.***
3. ***Objectives:*** ***The evaluation measures allowing us to select the most satisfactory one.***
4. ***Constraints:*** ***The technical requirements limiting the possible values of the decision variables.***

Constraints

- ***According to the studied problem, one can distinguish two classes of constraints: the endogenous constraints and the exogenic constraints.***
- ***The endogenous constraints represent some conditions directly related to the system to be optimized and to its performance (resource capacity, availability dates of resources, sequences of operations to be performed or ranges of products...).***
- ***The exogenic constraints are generally imposed by external element independently from the considered system (due dates imposed for each activity, priorities of some activities, priorities of some customers, tardiness permitted for some activities...).***

Criteria and objectives

- ***Among the set of feasible solutions, we have to select the most satisfactory one.***
- ***The notion of "satisfaction" depends on a criterion already defined.***
- ***The criteria represent some objective functions we can use to evaluate and to compare the performances of the feasible solutions.***
- ***The criteria can be conflicting: in this case, we consider the multiobjective/multicriteria optimization.***

Complexity and diversity

Complexity

- Illustration: let us consider the total tardiness minimization problem on a single machine on which a set of N independent jobs (activities) has to be performed. The problem consists to find a sequence of jobs with the aim of minimizing the total tardiness where every job is characterized by a processing time and a due date.
- Obviously, we have N possibilities for determining the job to be scheduled in the first position. Then, N-1 remaining jobs will be candidate to the second position. More generally, we will have (N-i+1) possible remaining jobs to put in the i-th position. Hence, we deduce that there are N.(N-1)...(N-i+1)...2.1 possible sequences (i.e., N! possibilities).
-To select one of these solutions, we need to measure the performance of each solution.
- Let suppose for example that N = 10 and we have a computer able in 0.1 *s* to explore the search space (constituted of the N! sequences), to evaluate these sequences and to select the best one. Moreover, let us assume that the same computer spends the same time for evaluating sequences of different values of N.

Complexity

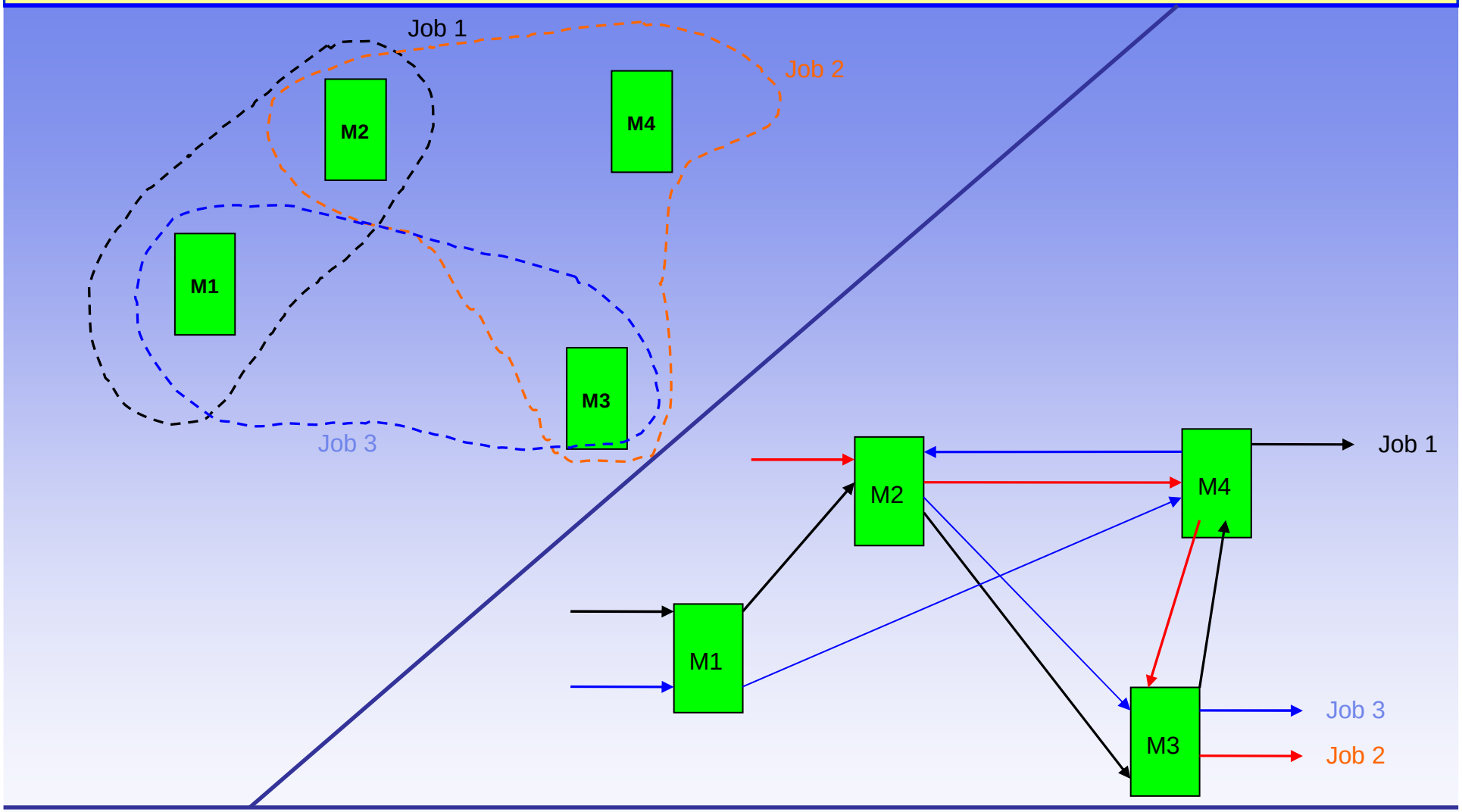
N	*Computation time*
15	**4 days**
20	**220 centuries**
25	**136 billions of years!**

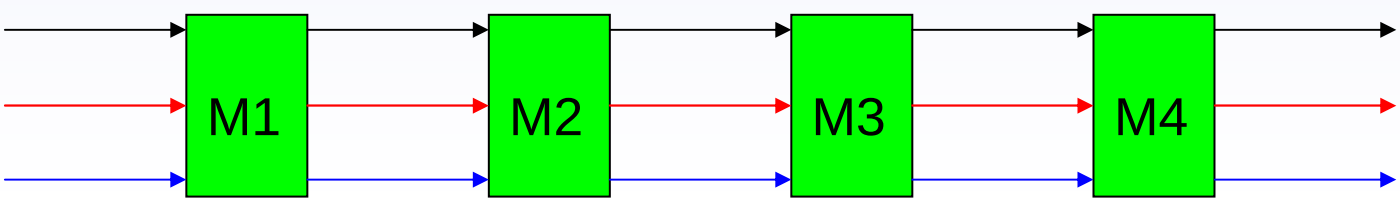
-This phenomenon is called the combinatorial explosion.
- Most of discrete optimization problems are combinatorial and belong to a special class of hard problem: the *NP-Hard* class.
- Most of researchers think we cannot construct polynomial time algorithms for solving problems of the *NP-Hard* class (see Garey & Johnson 1979).

Diversity

-The diversity leads to the specificity of any discrete optimization problem.
- Consequently, we cannot imagine a generic solution.
- Such a diversity is due to the existence of different and numerous industrial configurations and systems to optimize.
- Numerous classes of problems are related to different structures and types of systems.

Example: diversity of production systems





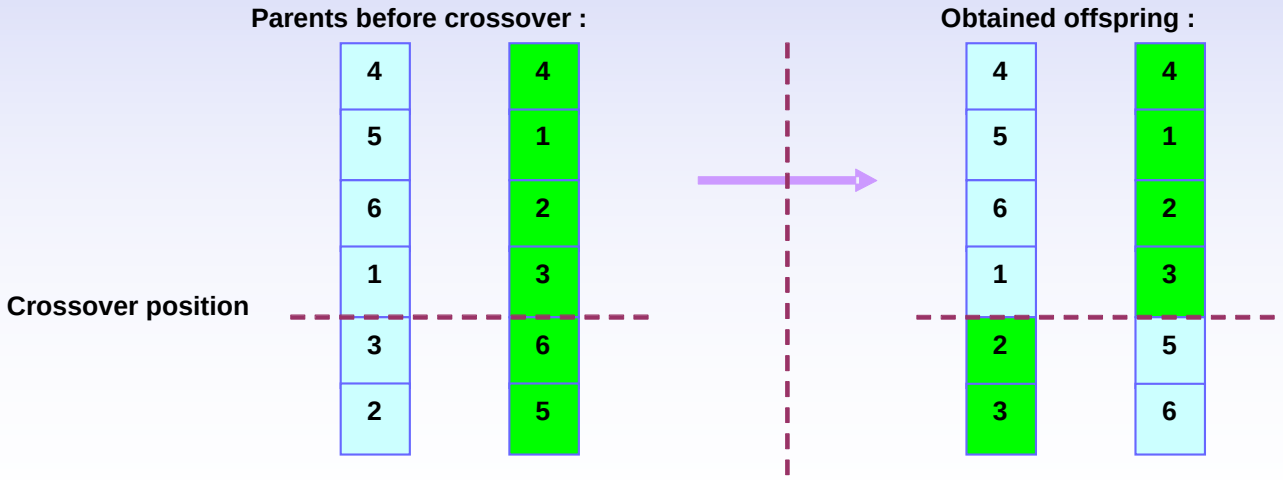
Used approaches

Heuristic Methods

- These methods cannot usually yield an optimal solution.
- They are fast and they can ensure a compromise between quality and computation time if they are rigorously implemented.
- The use of these methods can be recommended when the studied problem has a high level of hardness (for example, the *NP-Hard* problems in the strong sense).
- Any effective heuristic needs a minimum effort in its design (local-optimality properties).
- The results that we can obtain by a heuristic can be experimentally evaluated by comparing to the literature results.
- The performance of a heuristic can also be compared to some lower bounds and/or by establishing its worst-case performance analysis.
- Different types of heuristics exist and are widely-used.
    - constructive heuristic methods (based on some priority rules),
    - local-search methods based on the exploration of the neighbourhood of an existing solution (Tabu Search, Simulated Annealing...)
    - population-based methods consisting in iterative improvements of a set of solutions (Genetic Algorithms, Swarm Particle Optimization, Evolutionary Algorithms, Ant Colonies...).

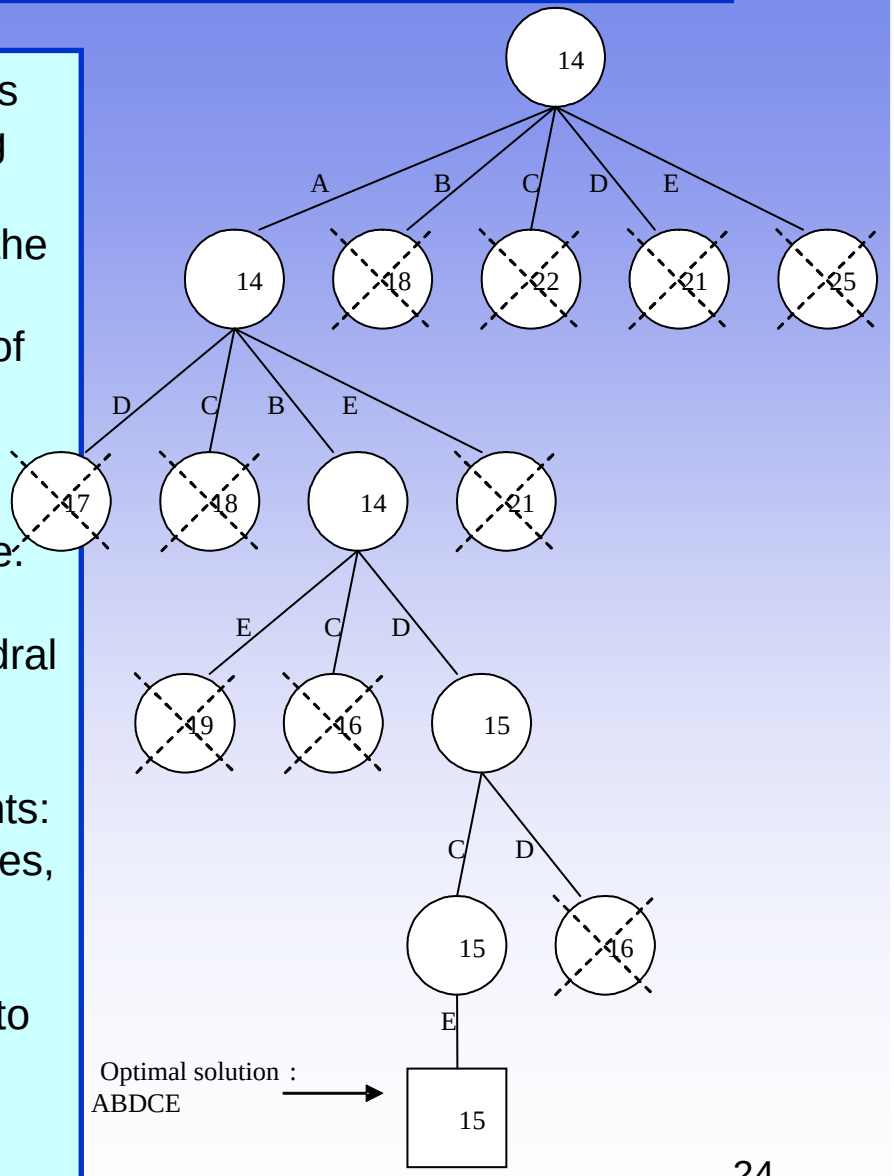
Heuristic Methods

- The disadvantage of these methods consists in the empirical performance of solution (we cannot evaluate the difference with the optimal solution).
- Another type of heuristics can overcome this disadvantage by yielding a guarantied performance for any instance of the problem.
- These heuristics can be obtained by using the polynomial approximation techniques and the performance analysis in the worst-case.
- This type of methods is generally very hard to construct and to analyze (see Pinedo).
- They allow us to simulate the system performance by fixing the decision variables in a simple and fast way.



Exact Methods

-when the structure of the problem is suitable it is possible to reach an optimal solution by applying an exact method.
- necessity of tools capable to explore implicitly the search space.
- this exploration allows us to reduce the space of visited solutions to a sub-space in which at least one optimal solution exists.
- we can discard the sub-spaces of dominated solutions in order to reduce the computation time.
- this technique can be applied for arborescent methods (branch-and-bound algorithms, polyhedral approaches, integer programming methods, dynamic programming…).
-It can be applied based on the following elements: heuristic solutions, lower bounds, dominance rules, valid inequalities, cuts, exploration strategies, recurrence relations...
-the elaborated exact methods allow us at least to improve the heuristic solutions when the optimal solution cannot be reached in reasonable computation time.

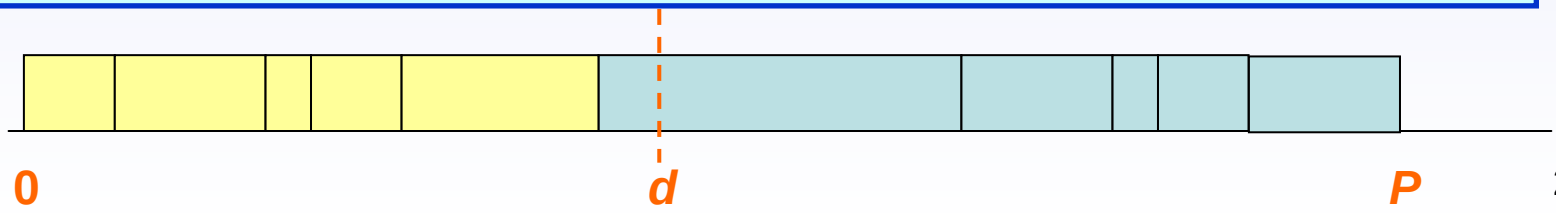


Example: Minimizing the weighted tardiness costs on a single production system under a common due date assumption

I. Kacem. Fully Polynomial-Time Approximation Scheme for the Weighted Total Tardiness Minimization with a Common Due Date.
Discrete Applied Mathematics (ELSEVIER), 2010, 158:9, 1035-1040.

Formulation

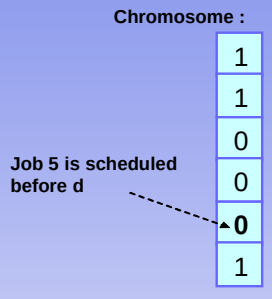
- The problem is to schedule n jobs/activities on a single machine (production system), with the aim of minimizing the total weighted tardiness cost of jobs.
- Every job i has a processing time pi and a weight wi (tardiness penalty).
- The machine is available at time 0 and can process at most one job at a time.
- The jobs have a common due date d.
- Without loss of generality, we consider that all the data are integers and that jobs are sorted in the WLPT order : $p1/w1 \geq p2/w2 \geq \ldots \geq pn/wn$.
- Due to the dominance of the WSPT order (the inverse one of WLPT), an optimal solution is composed of two sequences of jobs assigned before and after the due date. In the sequence before the due date any order is optimal (early jobs). In the sequence after the due date the jobs must be scheduled in the WSPT order (Smith).
- Assumption: d<P where P is the total processing time.
- The problem is NP-Hard in the ordinary sense.



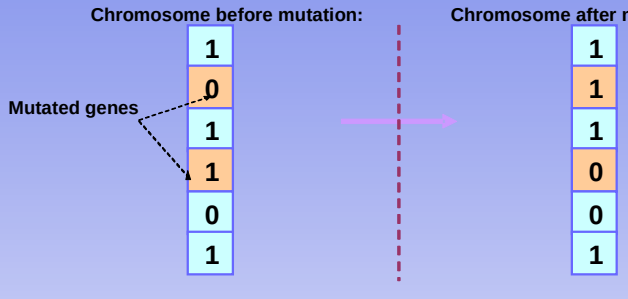
Some previous works

- This type of problems has been studied in the literature under various assumptions (Cheng 2005, Abdul-Razaq et al 1990, Lawler 1977, Potts & Van Wassenhove 1985).
- Fathi et al (1990, IIE Transactions): the problem has a 2-approximation algorithm.
- Kellerer & Strusevich (2006, TCS): dynamic programming in O(n.P.Ub^2) time.
- Kellerer & Strusevich (2006, TCS): FPTAS in O(n^6log(W)/(eps^3))
- This work:
  - New DP in O(n.d) time,
  - New FPTAS in O(n^2/eps) time.

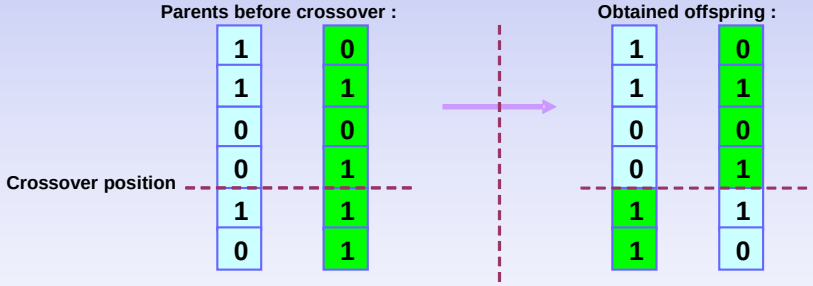
Example of a heuristic: a genetic algorithm



(*a*) *solution representation*



(*b*) *mutation operator*



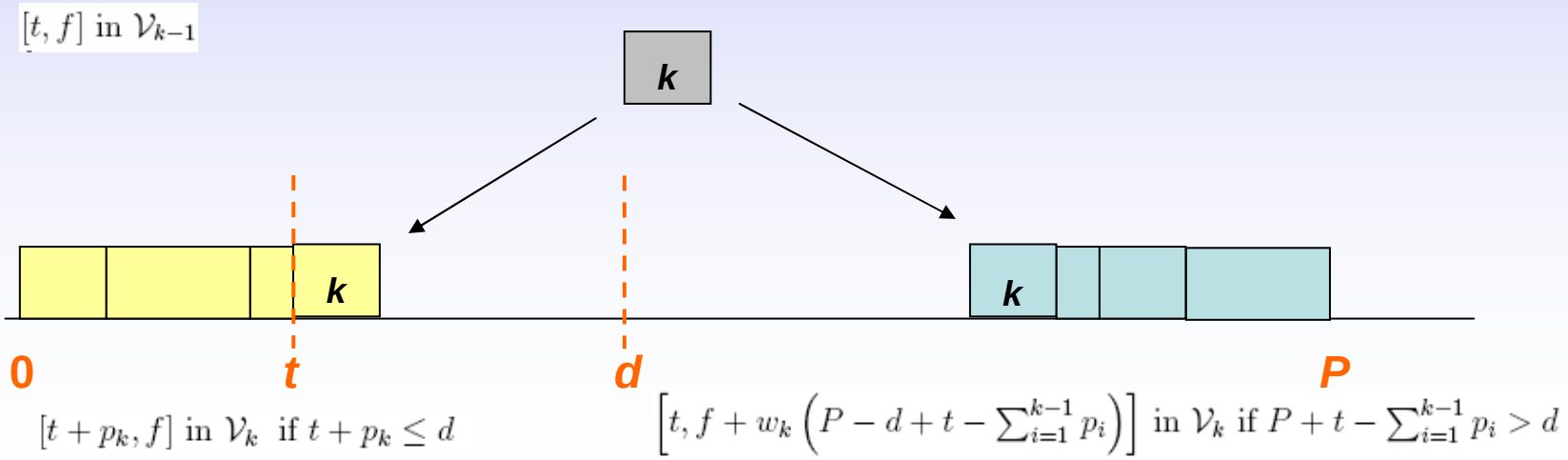
(*c*) *crossover operator*

The problem can be solved by a binary genetic algorithm.

In the chromosome, the value 0 is associated with an early job and the value 1 is associated with a tardy one.

Example of an exact method: New Dynamic Programming

The problem can be optimally solved by applying the following standard dynamic programming algorithm A. This algorithm generates iteratively some sets of states. At every iteration k, a set $\mathcal{V}_k$ composed of states is generated ($0 \leq k \leq n$). Each state $[t, f]$ in $\mathcal{V}_k$ can be associated to a special feasible schedule for the first k jobs. In this **special schedule**, the early jobs are scheduled starting from time 0 and the tardy jobs are scheduled so that they complete exactly at time P. Variable t denotes the completion time of the last early job (scheduled before d) and f is the weighted total tardiness of the corresponding schedule. This algorithm can be described as follows:



New Dynamic Programming

Algorithm A

(i). Set $\mathcal{V}_0 = \{[0,0]\}$.

(ii). For $k \in \{1, 2, ..., n\}$,

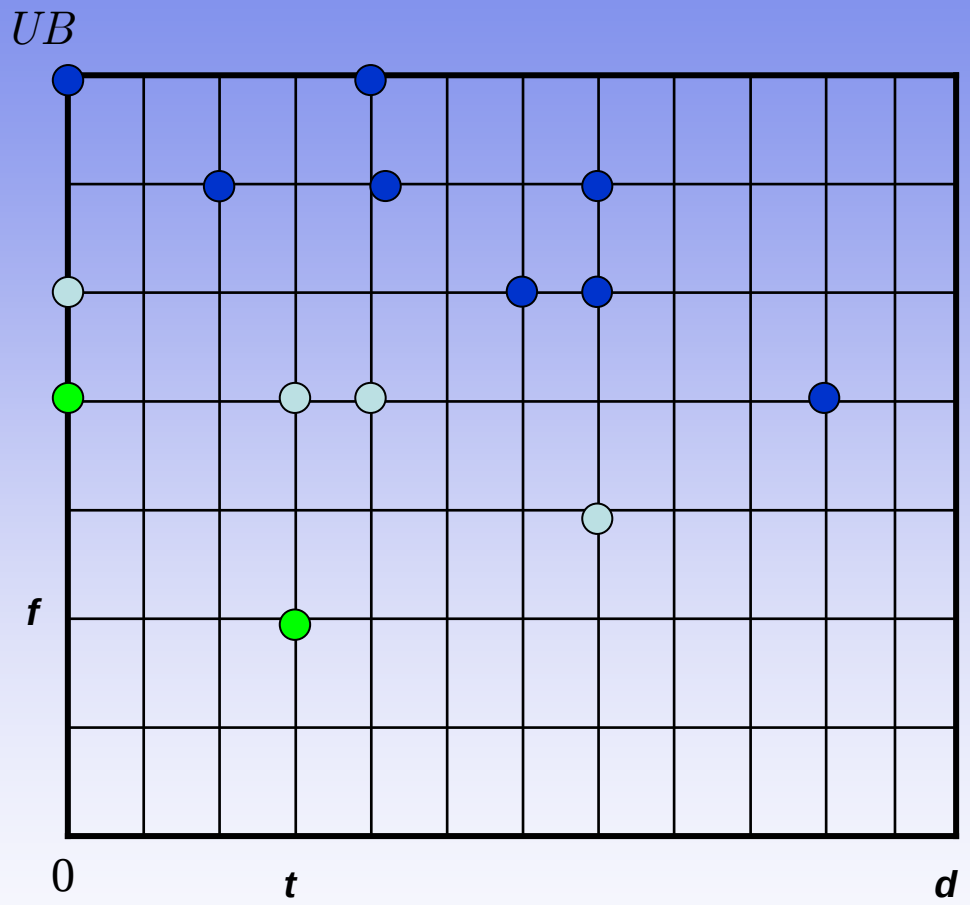
For every state $[t, f]$ in $\mathcal{V}_{k-1}$:
1) Put $[t + p_k, f]$ in $\mathcal{V}_k$ if $t + p_k \leq d$
2) Put $\left[t, f + w_k \left(P - d + t - \sum_{i=1}^{k-1} p_i\right)\right]$ in $\mathcal{V}_k$ if $P + t - \sum_{i=1}^{k-1} p_i > d$
Remove $\mathcal{V}_{k-1}$

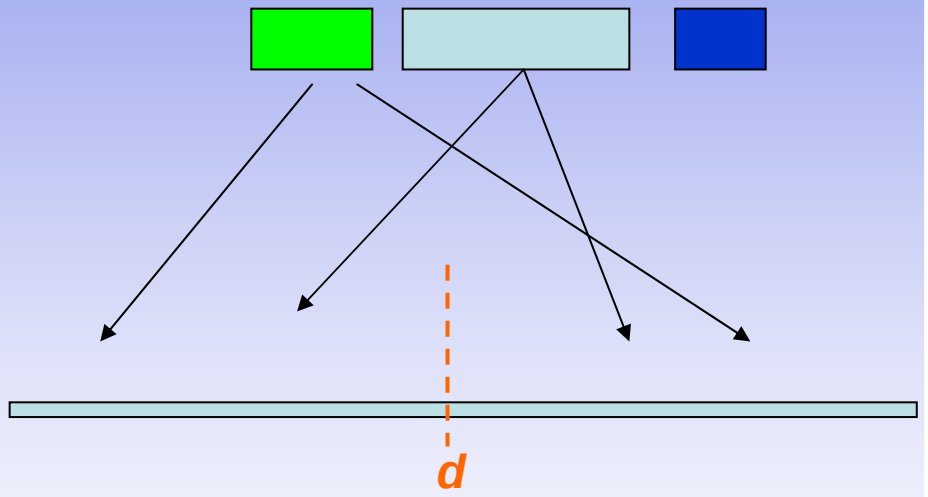
(iii). $\varphi^* (\mathcal{P}) = \min_{[t,f] \in \mathcal{V}_n} \{f\}$.

Let UB be an upper bound on the optimal weighted tardiness for problem $(\mathcal{P})$. If we add the restriction that for every state $[t, f]$ the relation $f \leq UB$ must hold, then the running time of A can be bounded by $ndUB$ (by keeping only one vector for each state $[t, f]$). Indeed, t and f are integers and at each iteration k, we have to create at most dUB states to construct $\mathcal{V}_k$. Moreover, the complexity of A is proportional to $\sum_{k=1}^{n} |\mathcal{V}_k|$.

However, this complexity can be reduced to $O(nd)$ by choosing at each iteration k and for every t the state $[t, f]$ with the smallest value of f (by keeping at most only one vector).

New Dynamic Programming: Illustration





Complexity of A: O(nd)

Fully Polynomial Time Approximation Scheme (FPTAS)

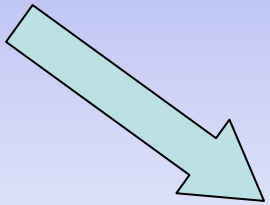
$$LB = \frac{\varphi_H(\mathcal{P})}{2},$$

$$q = \left\lceil \frac{2n}{\varepsilon} \right\rceil,$$

$$\delta = \frac{\varphi_H(\mathcal{P})}{q}$$

DEFINITION: Given ε>0, an FPTAS finds (1+ ε)-approximation with a time-complexity polynomial in (1/ ε) and in the input size.

PRINCIPLE: modification of the execution of an exact algorithm



We split the interval $[0, \varphi_H(\mathcal{P})]$ into q equal subintervals $I_r = [(r-1)\delta, r\delta]_{1 \le r \le q}$ of length δ. Our algorithm A'_ε generates reduced sets $\mathcal{V}_k^{\#}$ instead of sets $\mathcal{V}_k$. The algorithm can be described as follows:

Algorithm A'_ε

(i). Set $\mathcal{V}_0^{\#} = \{[0,0]\}$.

(ii). For $k \in \{1,2,3,...,n\}$,

For every state $[t,f]$ in $\mathcal{V}_{k-1}^{\#}$:
1) Put $[t+p_k, f]$ in $\mathcal{V}_k^{\#}$ if $t+p_k \le d$
2) Put $\left[t, f+w_k\left(P-d+t-\sum_{i=1}^{k-1} p_i\right)\right]$ in $\mathcal{V}_k^{\#}$ if $P+t-\sum_{i=1}^{k-1} p_i > d$

Remove $\mathcal{V}_{k-1}^{\#}$

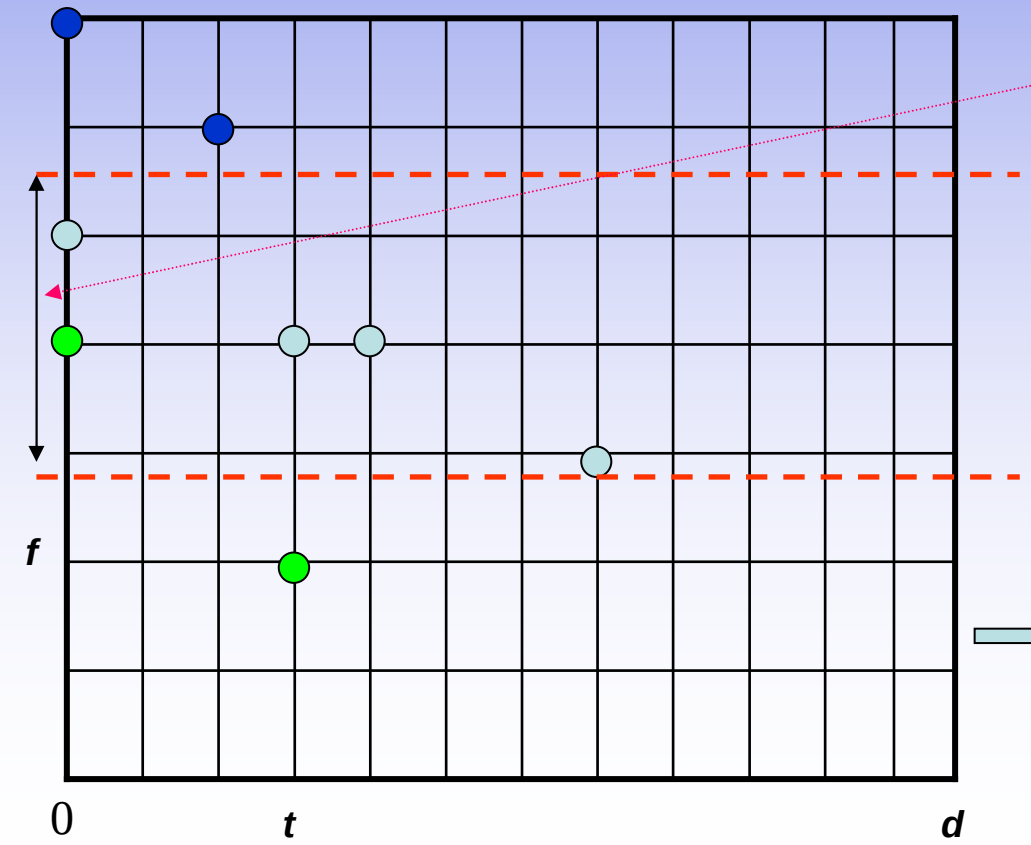
Let $[t,f]_r$ be the state in $\mathcal{V}_k^{\#}$ such that $f \in I_r$ with the smallest possible t (ties are broken by choosing the state of the smallest f).
Set $\mathcal{V}_k^{\#} = \{[t,f]_r \,|\, 1 \le r \le q\}$.

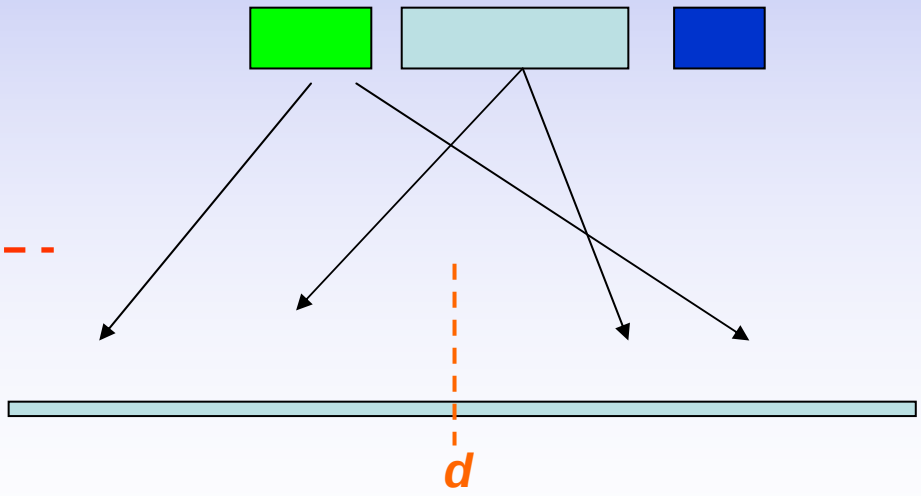
(iii). $\varphi_{A'_\varepsilon}(\mathcal{P}) = \min_{[t,f]\in\mathcal{V}_n^{\#}}\{f\}$.

$$LB = \frac{\varphi_H(\mathcal{P})}{2},$$

$$q = \left\lceil \frac{2n}{\varepsilon} \right\rceil,$$

$$\delta = \frac{\varphi_H(\mathcal{P})}{q}$$





Complexity of A′ε: O(n²/ε)

Conclusions

- The discrete optimization problems are the key of numerous management and economic issues in different business and service systems.
- Despite the hardness of such problems, it is usually very interesting to formulate, to understand and to solve them.
- The managers and researchers should continue to work together in order to progress on these important problems.
- Substantial gains can be reached in the financial and scientific fields.