General Problem Solvers for Combinatorial Optimization Problems by Metaheuristics

Toshihide Ibaraki

(with M. Yagiura and K. Nonobe)

Kyoto University

SIAM Conference on Optimization, Toronto, May 22, 2002

Outline of the talk

1. Combinatorial optimization problems
2. Standard problems and general problem solvers
3. Metaheuristics
4. Implementations and computational results for some applications
5. Future directions

Combinatorial optimization problems

minimize (maximize) $f(x)$

subject to $x \in F$

where feasible region F is combinatorial (discrete); e.g., a subset of $\{0,1\}^n$, a subset of Z^n, edge set E of a graph $G = (V, E)$, vertex set of G, the set of all permutations of n elements, a family of subsets of an n–set, the set of mappings from an n–set to an m–set, etc.

These are abundant in real world applications.

General problem solvers?

- Many combinatorial problems are difficult to solve (e.g., NP-hard) and need time to develop effective algorithms.

  ⇒ General problem solvers are necessary.
- In this direction, theory tells that all problems in NP can be reduced to an NP-hard problem A.

  ⇒ An algorithm for A can be used as a general problem solver.
- The NP-hard problem A is difficult to solve exactly.

  ⇒ Approximate algorithm for A.

- The reductions between NP problems may blow up the sizes.
- The reductions may not preserve the metric of objective functions. (A good solution of A may not be a good solution of the target problem.)

$\Rightarrow$ Natural reductions are desirable.

- Different types of standard problems A_1, A_2, …, A_k must be prepared.

⇒ The problem instance at hand is formulated as an instance of an appropriate standard problem A_i, and is then solved by an algorithm for A_i..

Our list of standard problems

- Integer programming problem (IP); Commercially available.
- (Weighted) constraint satisfaction problem (CSP, WCSP)
- Maximum satisfiability problem (MAX SAT)
- Set covering problem (SCP)
- Generalized assignment problem (GAP)
- Generalized quadratic assignment problem (GQAP)
- Resource constrained project scheduling problem (RCPSP)
- Vehicle routing problem (VRP)
- Cutting stock problem (CSTP)
- 2-Dimensional Packing Problem (2PP)
- …

- Each standard problem A_i must be **as general as possible**, while maintaining its special structure that allows a specialized solution strategy.

● Algorithms for standard problems must be

— efficient in the practical sense,

— robust against small structural changes in the problems,

— easy to apply.

⇓

Metaheuristics

— simulated annealing,

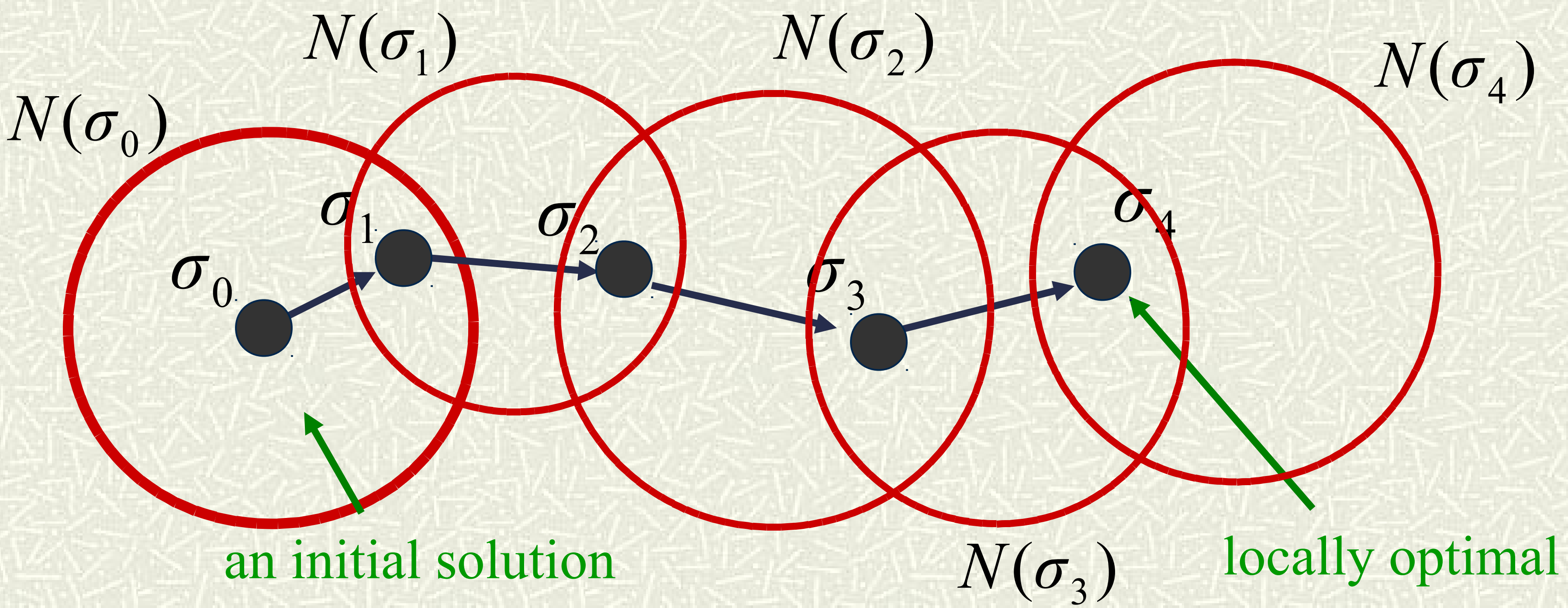
— genetic algorithms,

— iterated local search,

— tabu search,

— others

Local search（LS） repeats replacing σ with a better solution in its neighborhood $N(\sigma)$



General framework of metaheuristics

Repeat the following steps until a convergence criterion is satisfied.

Step 1: Generate an **initial solution** (based on the computational history so far).

Step 2: Apply (generalized) **local search** to find a good locally optimal solution.

Step 1 -- random generation, mutation, cross-over operation, path relinking, …, from population of good solutions obtained so far.

Step 2 -- simple local search, random moves with controlled probability, best moves with a tabu list, local search with modified objective functions (e.g., with penalty of infeasibility), …

Our list of standard problems

- Integer programming problem (IP); Commercially available.
- (Weighted) constraint satisfaction problem (CSP, WCSP)
- Maximum satisfiability problem (MAX SAT)
- Set covering problem (SCP)
- Generalized assignment problem (GAP)
- Generalized quadratic assignment problem (GQAP)
- Resource constrained project scheduling problem (RCPSP)
- Vehicle routing problem (VRP)
- Cutting stock problem (CSTP)
- 2-Dimensional Packing Problem (2PP)
- …

Resource constrained project scheduling problem (RCPSP)

- Activities $j = 1, 2, \ldots, n$.
- Resources $r = 1, 2, \ldots, R$ and $s = 1, 2, \ldots, S$.

-- renewable resources (machine, manpower, etc.): $K_{r,t}$ available in each period t,

-- nonrenewable resources (budget, raw materials, etc.): K_s available in total.

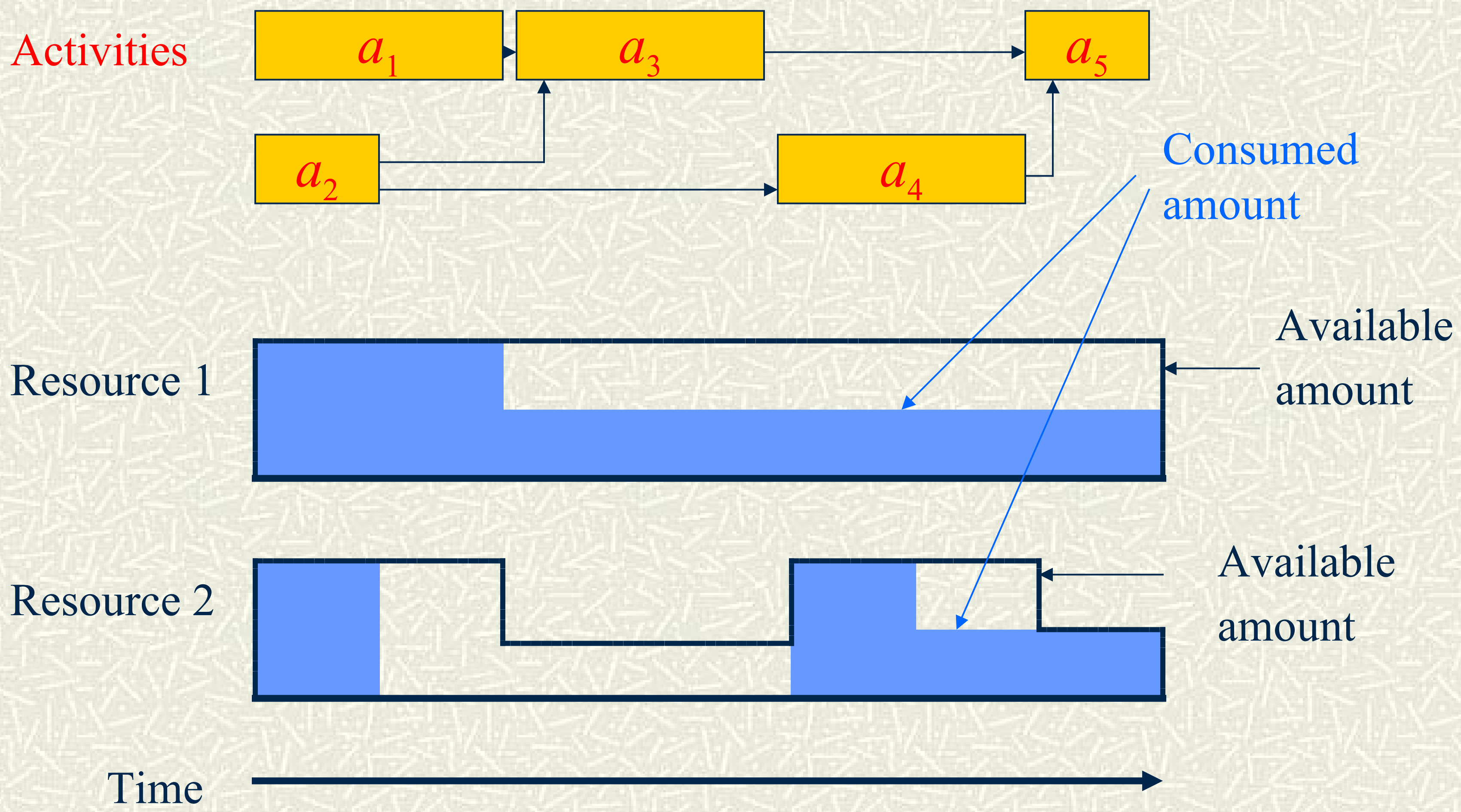
- Process mode m of each activity can be chosen.

-- processing time p_m,

-- renewable resources $k_{r,m,t}$ in the t-th period after start,

-- nonrenewable resources $k_{s,m}$.

RCPSP


Activities
a_1
a_3
a_5
a_2
a_4
Consumed amount
Available amount
Resource 1
Available amount
Resource 2
Time

Constraints and objectives to be included

- Precedence constraints between activities.
- Objective functions to minimize
  -- makespan, weighted sum of delays, …
- Setup activities.
- Other constraints on modes, start times, completion times and/or processing times.
- Schedules to minimize the weighted sum of penalties on constraints,
  -- hard and soft constraints.

Implementation of RCPSP

- Tabu search.
- Solutions are encoded as (m, π), where m is the modes of activities and π is a permutation of all activities.
- Heuristic algorithm to construct a schedule from (m, π), which satisfies all hard constraints.
- Reduced neighborhood obtained from the critical path analysis of the current schedule.
- Automatic control of the tabu tenure.

Computational experiment

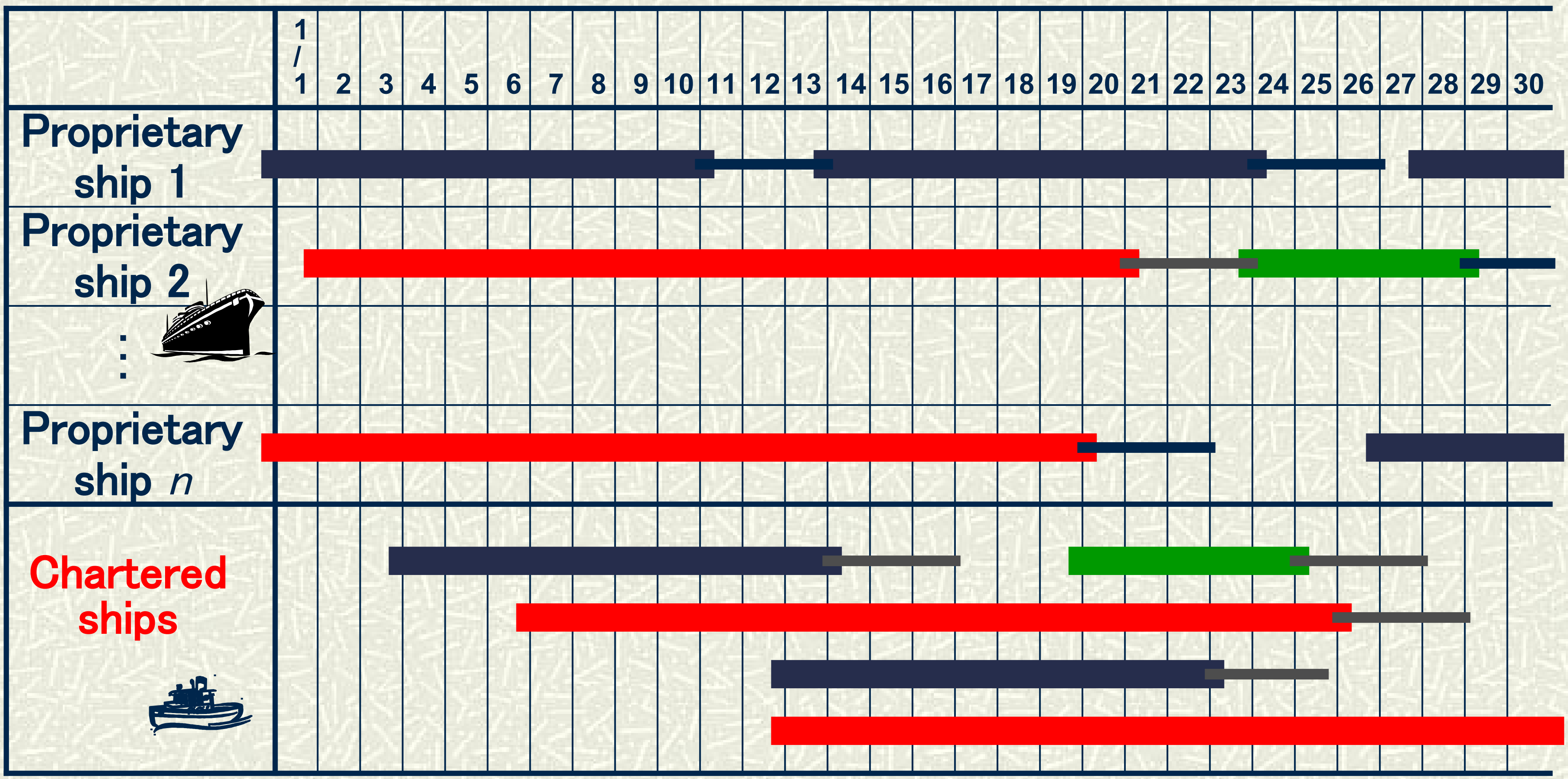
- Job shop scheduling
- Benchmark problems in PSPLIB
- Problems from real applications.
- …

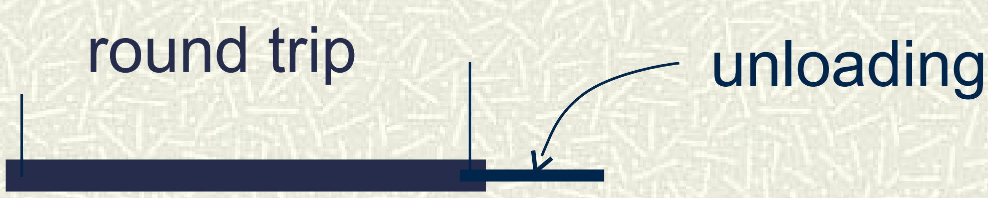
Ship scheduling

- Transportation of natural resources;
  e.g., oil, iron ore, etc.
- Activities: trips of given origins, destinations, and amounts of resources.
- Two types of ships: proprietary and chartered.
- Constraint on the storage level at the yard (tank).
- Objective: Minimization of the number of chartered trips.

Ship Scheduling

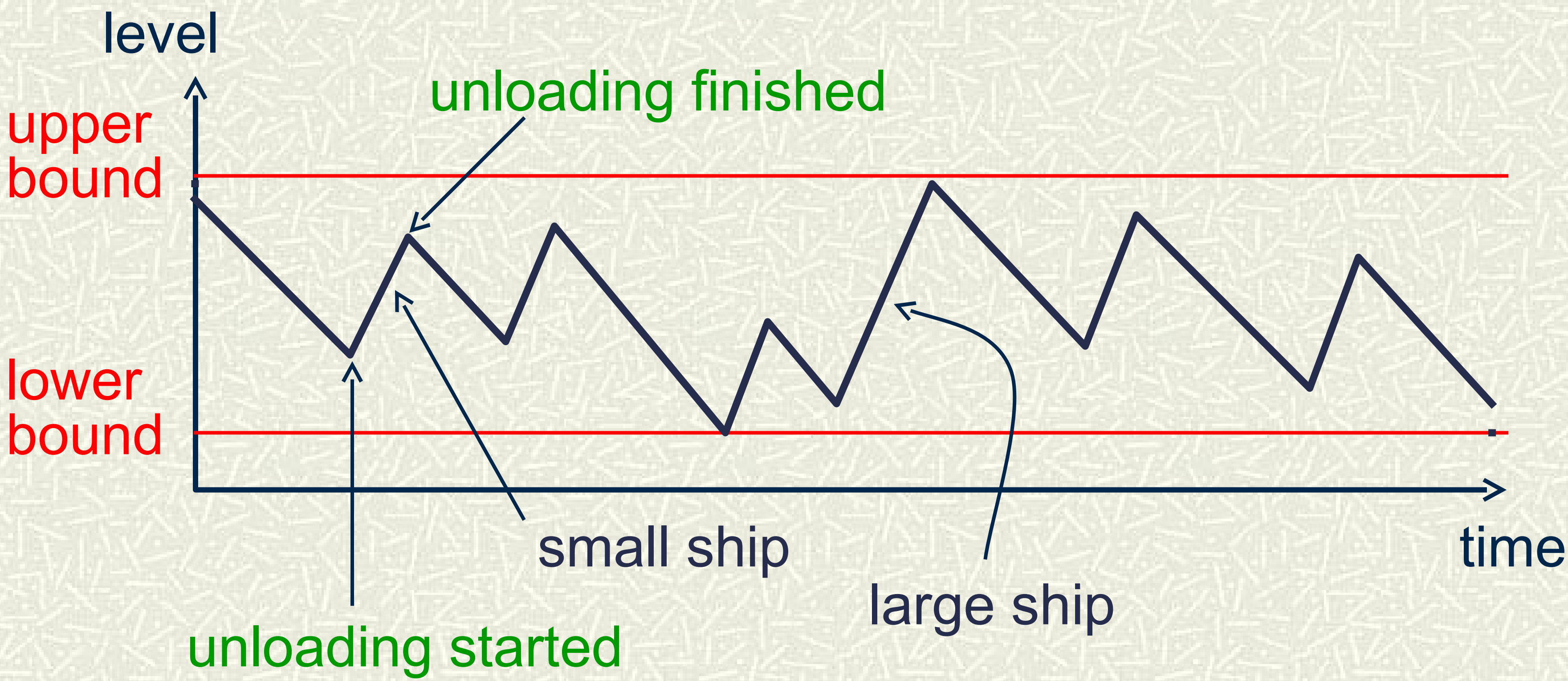
Schedule table





Constraint on the levels of tanks (yards)

- Given: Consumption rate from each tank (yard).
- Capacities of tanks (yards): upper and lower bounds.



Typical instances

- # activities: about 100.
- Scheduling horizon: 1 year
- # proprietary ships: 4
- # chartered ships: 5
- # destinations: 4

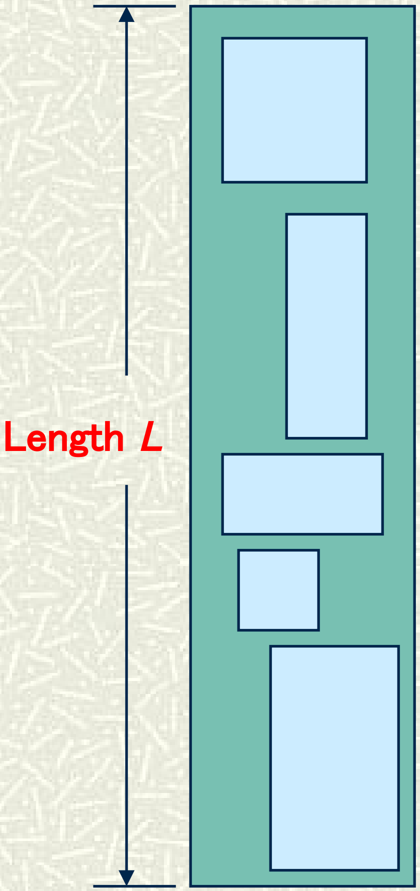
Computation time

- Succeeded to obtain practically useful schedules in about 10 minutes on SUN SPARC ULTRA 2.

Work space scheduling of large construction parts

––– 2-dimensional packing of activities –––

- Large construction parts (ships, bridges, etc.) in a narrow factory.
- Each part occupies some area, and stays in the same place until completion. (Then it is removed by a crane.)
- Each part has its ready time, deadline, and processing time. Required resources change with time.



Solution strategy

- **1st stage:** After discounting L to σL (e.g., $\sigma = 0.9$), apply RCPSP in the direction of t under the resource amount of σL..

1st

- **2nd stage:** Apply RCPSP in the direction of L, assuming that each day has unit amount of its own resource (each activity consumes the resources of the scheduled days).

2nd

Our list of standard problems

- Integer programming problem (IP); Commercially available.
- (Weighted) constraint satisfaction problem (CSP, WCSP)
- Maximum satisfiability problem (MAX SAT)
- Set covering problem (SCP)
- Generalized assignment problem (GAP)
- Generalized quadratic assignment problem (GQAP)
- Resource constrained project scheduling problem (RCPSP)
- Vehicle routing problem (VRP)
- Cutting stock problem (CSTP)
- 2-Dimensional Packing Problem (2PP)
- …

2-Dimensional packing problem

Input: A set of rectangles $I = \{1,2,\ldots,n\}$;
each i has several modes.
mode: (width, height, spatial cost functions)
Output: Modes and x, y coordinates of all rectangles.

It is asked to place all rectangles in the plane without overlap so that the objective function is minimized.

Mathematical formulation

Input: **Modes** $(w_i^{(k)}, h_i^{(k)}, p_i^{(k)}(x_i), q_i^{(k)}(y_i))$ **for**
$i \in \{1,2,\ldots,n\}$ **and** $k \in M_i$,
objective functions : g **and** c.

Output : **A solution** π, i.e.,
packing $(x_i(\pi), y_i(\pi))$ **(lower left corner),**
mode $\mu_i(\pi)$, **for all** $i \in \{1,2,\ldots,n\}$.

$p_{\max}(\pi) = \max_i p_i^{(\mu_i(\pi))}(x_i(\pi))$ (s imilarly, $q_{\max}(\pi)$)

minimize $g(p_{\max}(\pi), q_{\max}(\pi)) + c(\mu(\pi))$

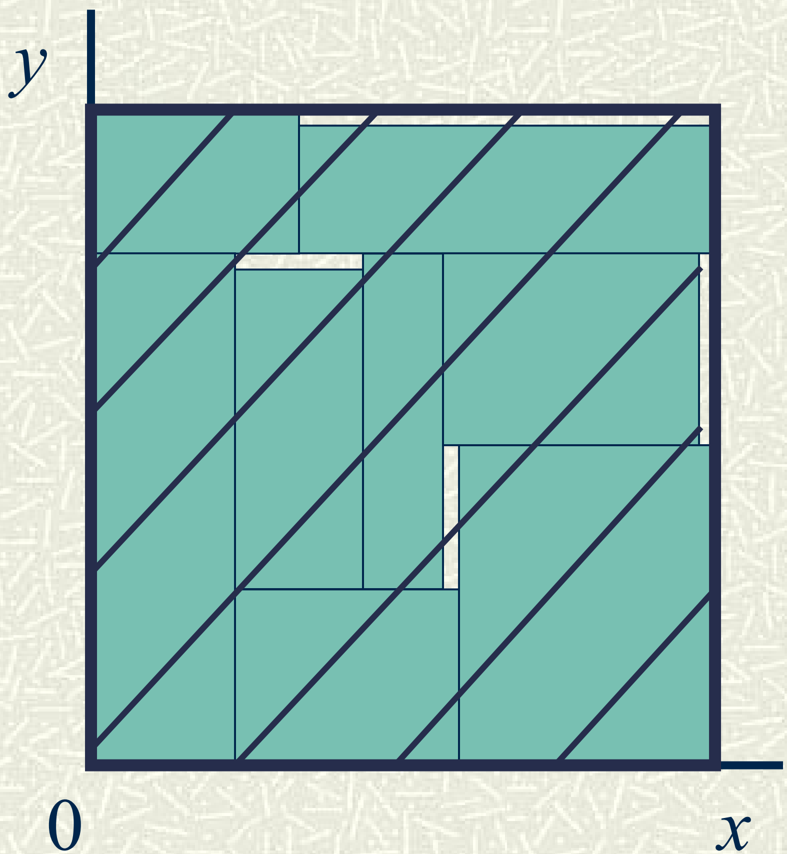
subject to : **no overlapping**

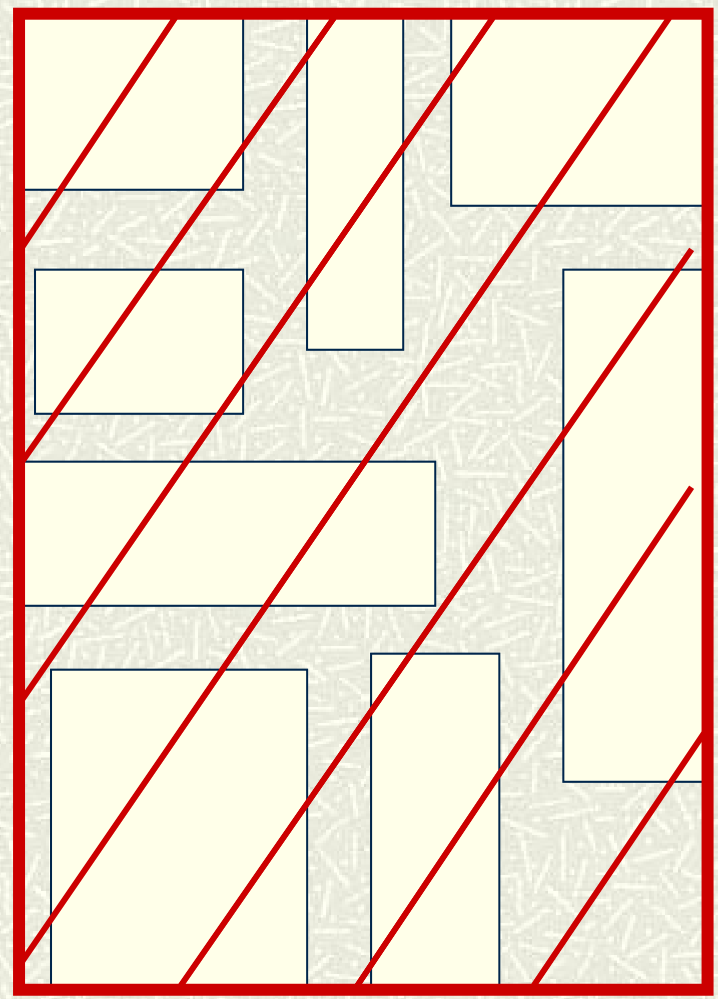
Functions $p_i^{(k)}(x_i)$ and $q_i^{(k)}(y_i)$ can be very general:

These can be nonconvex, noncontinuous as long as piecewise linear.

Objective function $g(p_{\max}(\pi), q_{\max}(\pi))$ is nondecreasing in $p_{\max}(\pi)$ and $q_{\max}(\pi)$.

Example 1 -- smallest area packing --



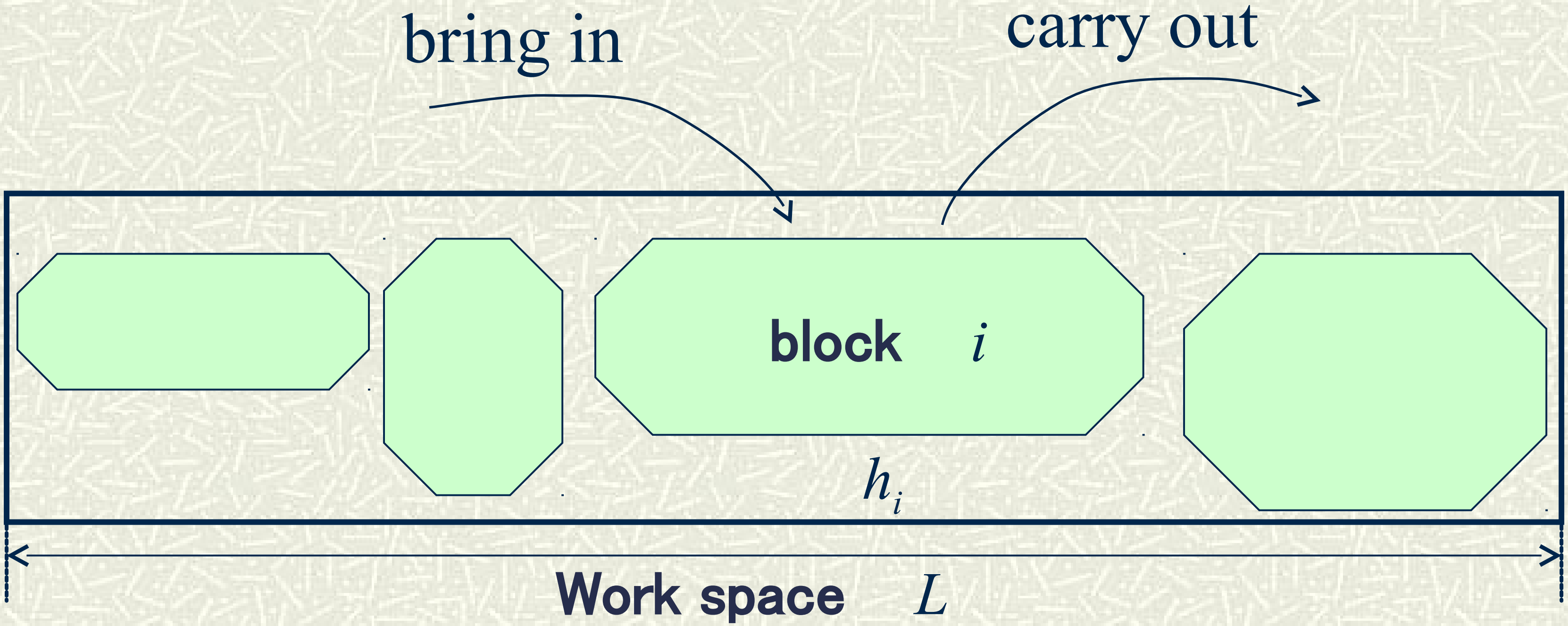


$$p_i(x_i) = \begin{cases} x_i + w_i, & x_i \geq 0 \\ +\infty, & x_i < 0 \end{cases}$$

Example 2 -- scheduling problem --

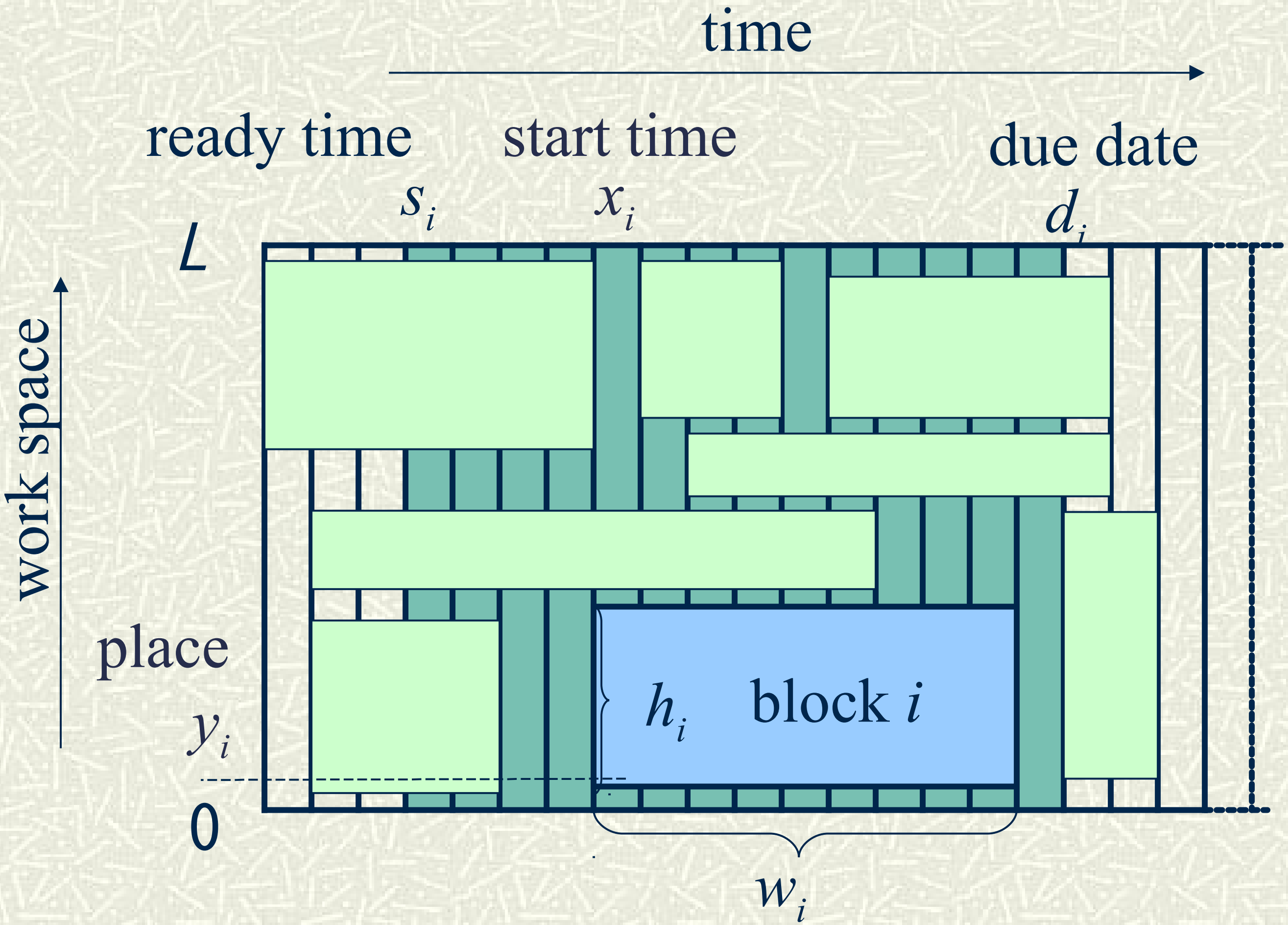
Building block $i \in \{1, 2, \ldots, n\}$:

length h_i, **processing time** w_i, **ready time** s_i, **due date** d_i

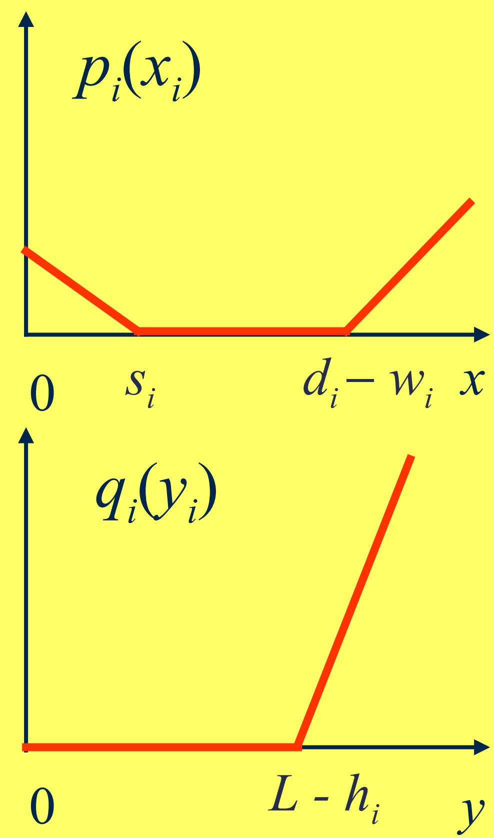


Determine the place and the start time of each block.

- **Rectangle: (processing time) × (length)**



cost function



objective function

$g(p, q) = p + q$

Representation of solutions

Direct search of rectangle locations is not appropriate, because there are uncountably many solutions.

Sequence pair (Murata, Nakatake, Fujiyoshi and Kajitani (1995)): Relative nonoverlapping positions are specified by a sequence pair of rectangles.

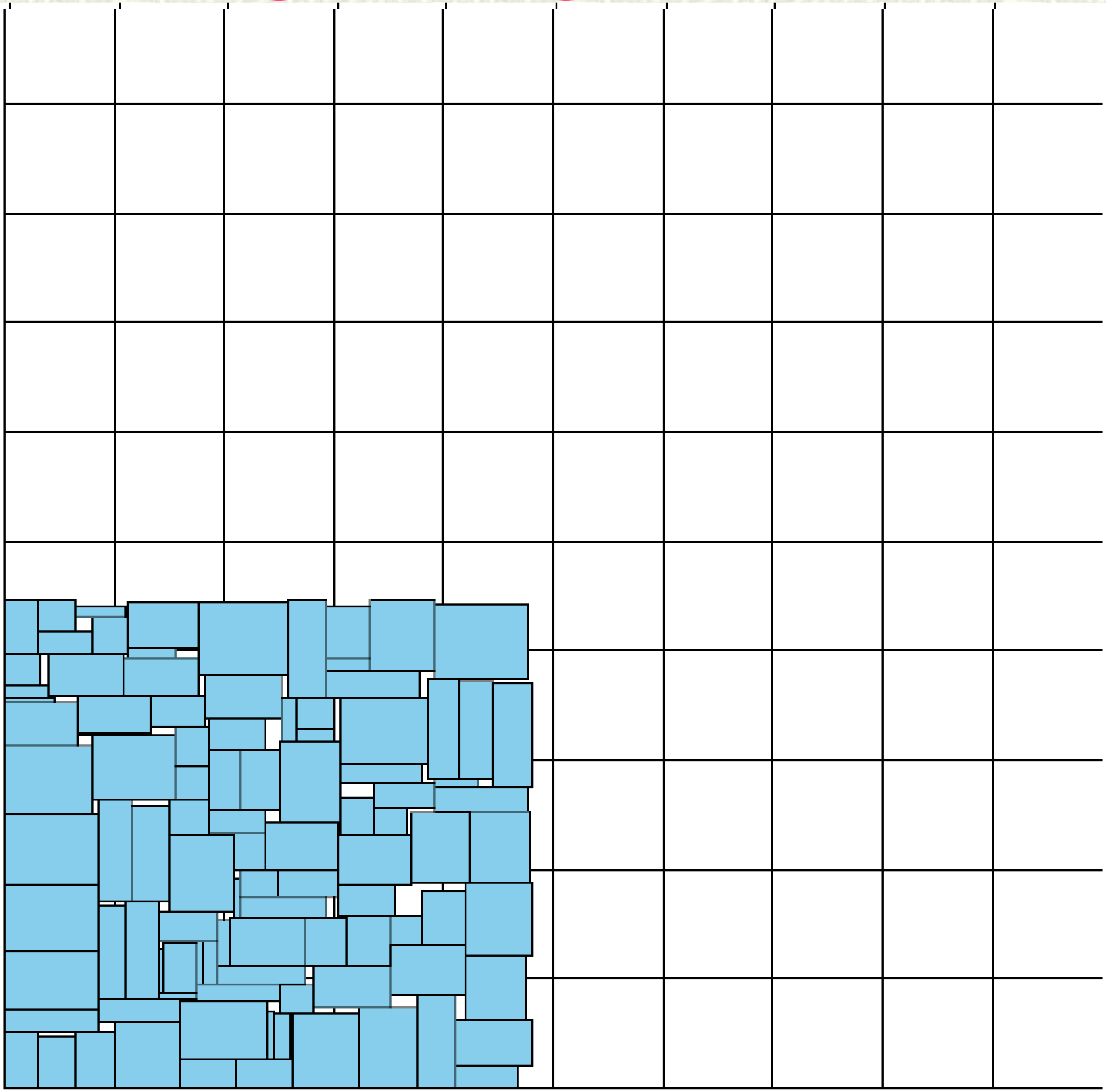
Search strategy of solutions

1. Local search is applied to search good sequence pairs and modes of rectangles.

   Shift, swap and change mode neighborhood s reduced by critical path ideas.

2. Given a sequence pair, exact optimal locations are computed by dynamic programming algorithm in polynomial time.

Packing rectangles into a small area



Conclusion and discussion

- Further improvement of metaheuristic algorithms.
- Increasing the formulation power of standard problems.
- Other standard problems.
- User interfaces. Supports for modeling the problems in applications.