A GRASP for Job Shop Scheduling

Maurício G. C. Resende

AT&T Labs Research

Florham Park, New Jersey

mgcr@research.att.com

http://www.research.att.com/~mgcr

May 1997

Joint work with S. Binato, W. J. Hery, & D. M. Loewenstern.

Agenda

- Job shop scheduling (JSS) problem
- GRASP for JSS
  - construction method
  - local search
- Computational experience
- Future directions & conclusions

Job shop scheduling

- schedule a set of jobs on a set of machines, such that
  - each job has a specified processing order on the set of machines
  - machines can process only one job at a time
  - each job has a specified duration on each machine
  - machine must finish processing job before it can begin processing another job (no preemption allowed)

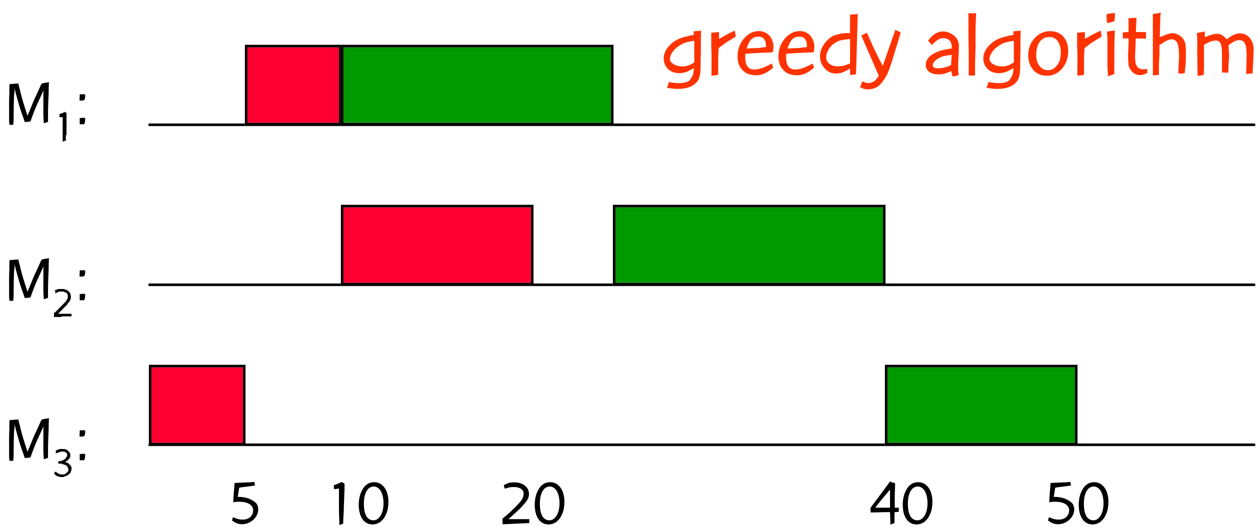
Job shop scheduling

- objective: minimize the maximum completion time (makespan) of jobs

example:

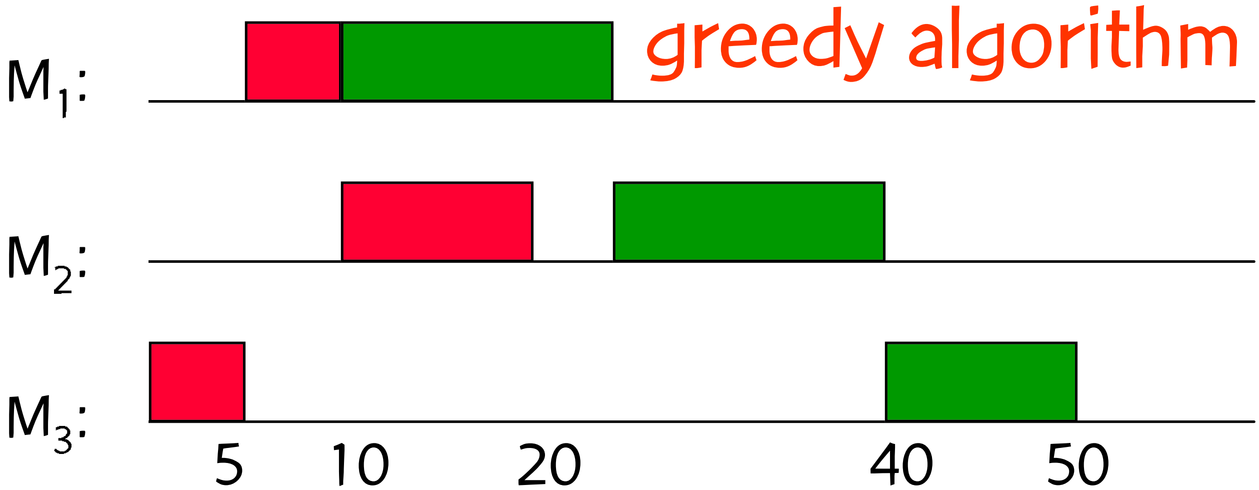
J_1: $M_1(15)$, $M_2(15)$, $M_3(10)$

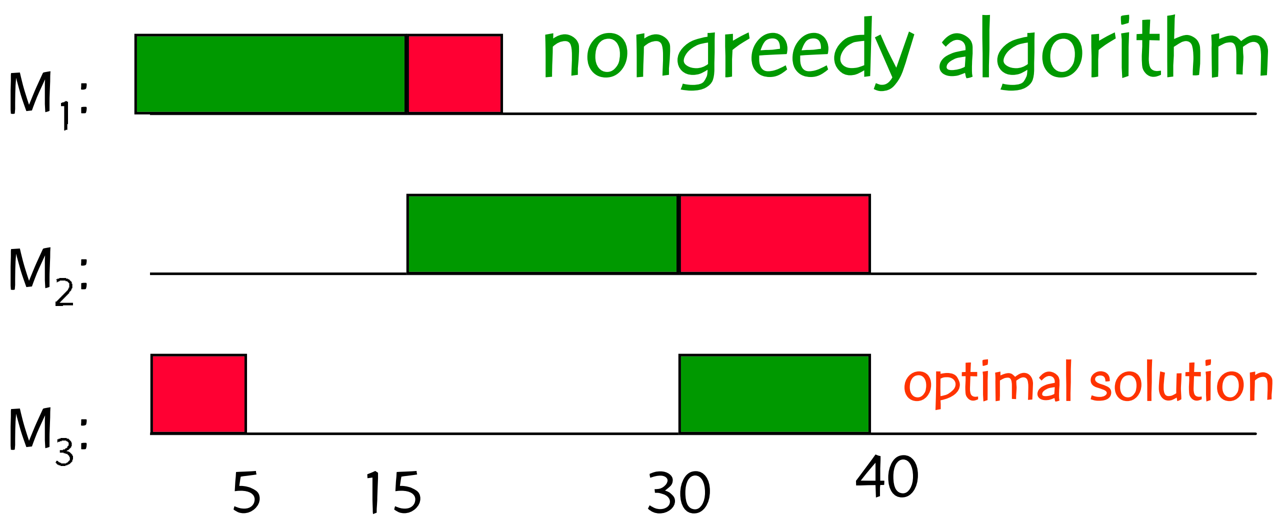
J_2: $M_3(5)$, $M_1(5)$, $M_2(10)$



J_1: $M_1(15)$, $M_2(15)$,$M_3(10)$

J_2: $M_3(5)$, $M_1(5)$, $M_2(10)$





Job shop scheduling

- NP-hard, even for
  - 2 machines with at most 3 operations per job and for 3 machines with at most 2 operations per job (Lenstra *et al.*, 1977; Gonzalez & Sahni, 1978)
  - 3 machines & unit processing times (Lenstra & Rinnooy Kan, 1979)
  - 3 jobs (Sotskov, 1991)
  - preemption allowed (Gonzalez & Sahni, 1978)

Solution approaches

- Exact methods (e.g. Applegate & Cook, 1991) solve MIP problem using:
  - lower bounds
  - polyhedral techniques
  - branching schemes

Solution approaches

- Approximate methods, e.g.
  - list schedules (use dispatching rules)
    - Lawrence, 1984
  - simulated annealing
    - van Laarhoven *et al.*, 1992
  - tabu search
    - Widmer, 1989
    - Dell'Amico & Trubian, 1993
    - Taillard, 1994
  - genetic algorithms
    - Dellacroce *et al.*, 1992
    - Kobayashi *et al.*, 1995

GRASP (greedy randomized adaptive search procedure)

- iteratively
  - samples solution space using a greedy probabilistic bias to construct a feasible solution
  - applies local search to attempt to improve upon the constructed solution
- keeps track of the best solution found

GRASP

```
best_obj = BIG;
repeat many times{
   x = grasp_construction( );
   x = local_search(x);
   if ( obj_function(x) < best_obj ){
         x* = x;
         best_obj = obj_function(x);
   }
}
```

bias towards greediness

good diverse solutions

Construction

- Construction is done one element at a time:
  - greedy construction
    - each candidate element is evaluated by a greedy function
    - element with the best evaluation is chosen
  - random construction
    - each candidate element is assigned an equal probability of being selected
    - one of these elements is chosen at random

Construction

- What is good about:
  - greedy construction
    - good quality solutions
    - local search quickly converges to local optimum
  - random construction
    - diverse solutions are generated
    - span solution space

Construction

- What is bad about:
  - greedy construction
    - little or no diversification
    - solution are usually sub-optimal
    - local search rarely converges to globally optimal solution
  - random construction
    - solutions are of poor quality
    - local search is slow to converge to local optimum

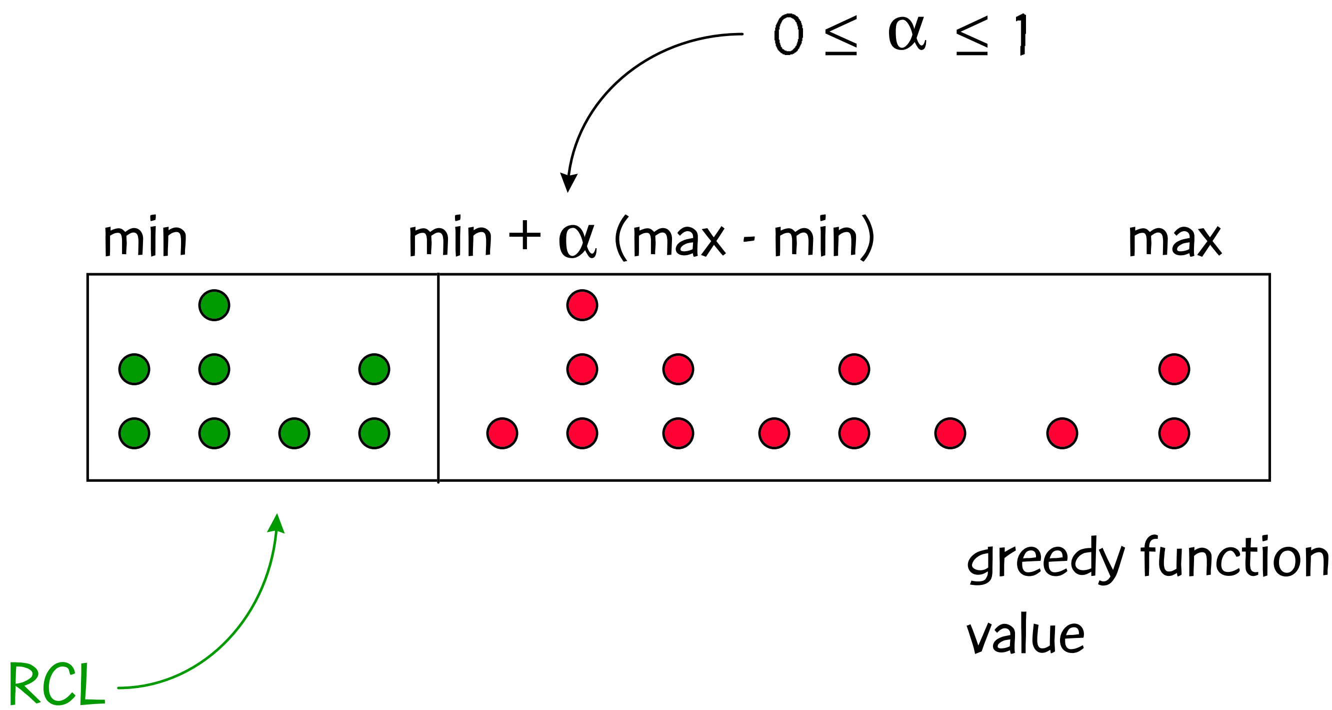
GRASP construction

- Tries to capture good features of
  - greedy construction
    - good quality solutions
    - fast local search convergence
  - random construction
    - diversification
- Tries to avoid bad features of
  - greedy construction
    - little or no diversification
  - random construction
    - bad quality solutions
    - slow local search convergence

GRASP construction

- repeat until solution is constructed
  - For each candidate element
    - apply a greedy function to element
  - Rank all elements according to their greedy function values
  - Place well-ranked elements in a restricted candidate list (RCL)
  - Select an element from the RCL at random & add it to the solution

GRASP construction



Local search

- There is no guarantee that constructed solutions are locally optimal w.r.t. simple neighborhood definitions.
- It is usually beneficial to apply a local search algorithm to find a locally optimal solution.

Local search

- Let
  - N(x) be set of solutions in the neighborhood of solution x.
  - f(x) be the objective function value of solution x.
  - x^0 be an initial feasible solution built by the construction procedure
- Local search to find local minimum

```
while ( there exists y ε N(x) | f(y) < f(x) ){
      x = y;
}
```

GRASP for JSS

- To define the GRASP, we need to specify
  - construction mechanism
  - greedy function
  - candidate list restriction parameter α
  - local neighborhood structure
  - local search algorithm

Construction mechanism

- Each job has a set of operations to be scheduled
- Schedules are constructed one operation at a time
- At each step of the construction, the candidates are operations that can be feasibly scheduled

Construction mechanism

- k-th operation of job j, is scheduled at a time that is the max of
  - completion time of (k-1)-th operation of job j
  - completion time of latest job to be processed on same machine as k-th operation of job j.

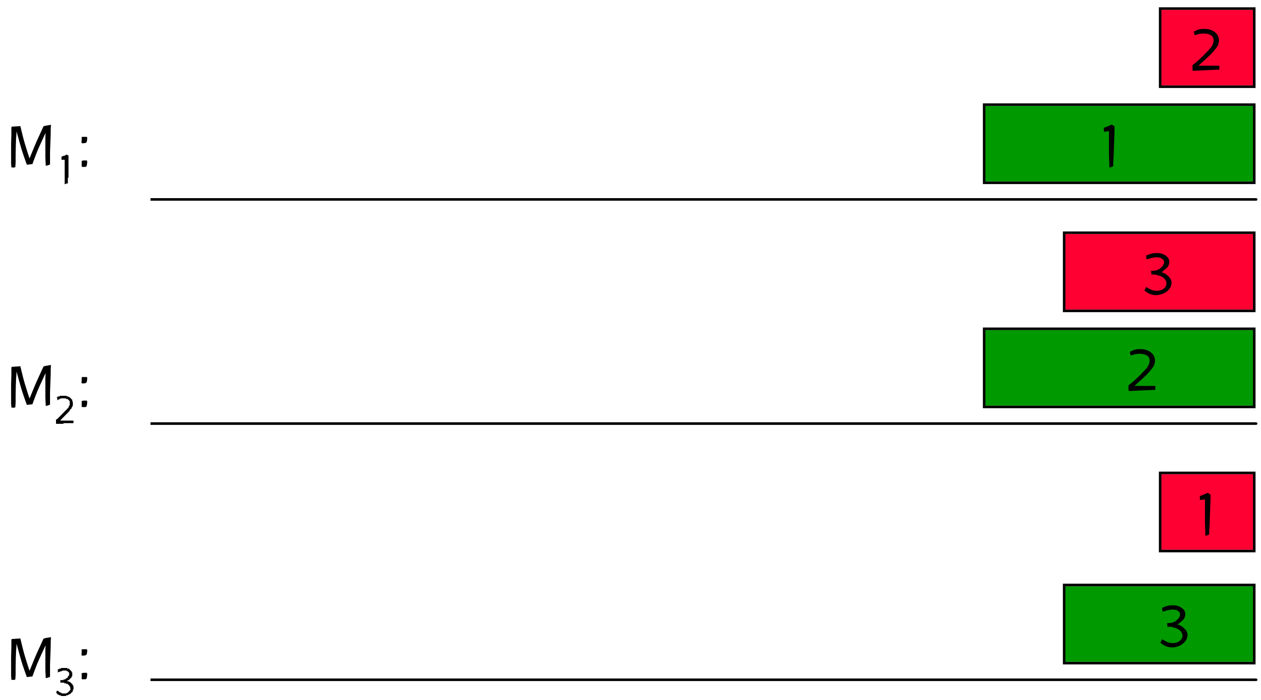
Greedy function

- For all candidate operations *o*, apply greedy function greedy(*o*):

  maximum{
  - maximum completion time of all previously scheduled operations
  - completion time of operation *o*, if operation *o* were to be next scheduled

  }

Greedy function



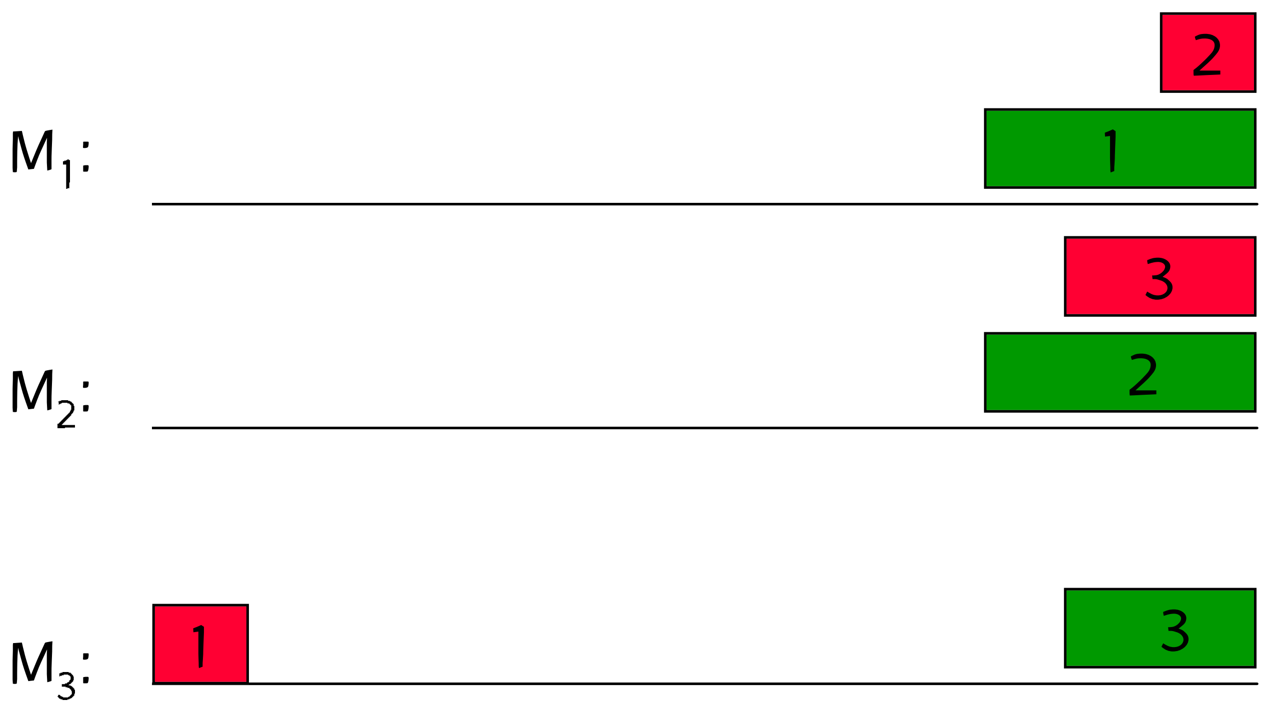
candidates = { 1 , 1 }

greedy (1) = 5; greedy (1) = 15

greedy choice = 1

suppose we take the greedy choice

Greedy function



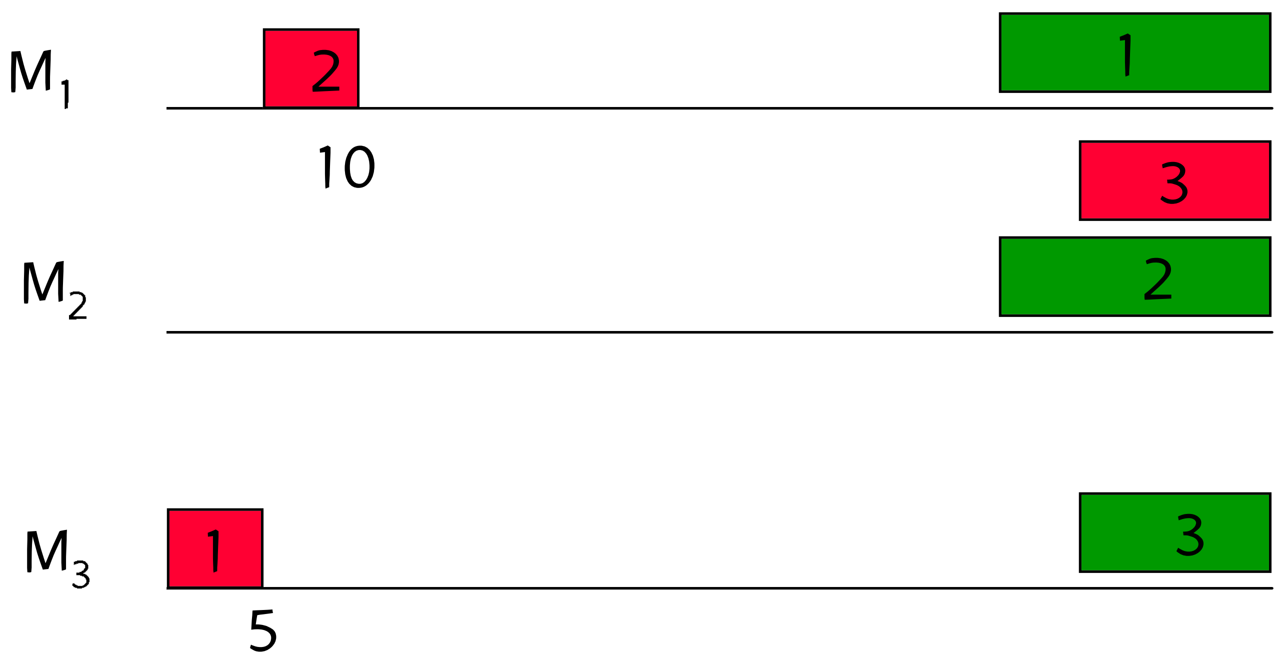
candidates = { 2 , 1 }

greedy (2) = 10; greedy (1) = 15

greedy choice = 2

suppose we take the greedy choice

Greedy function

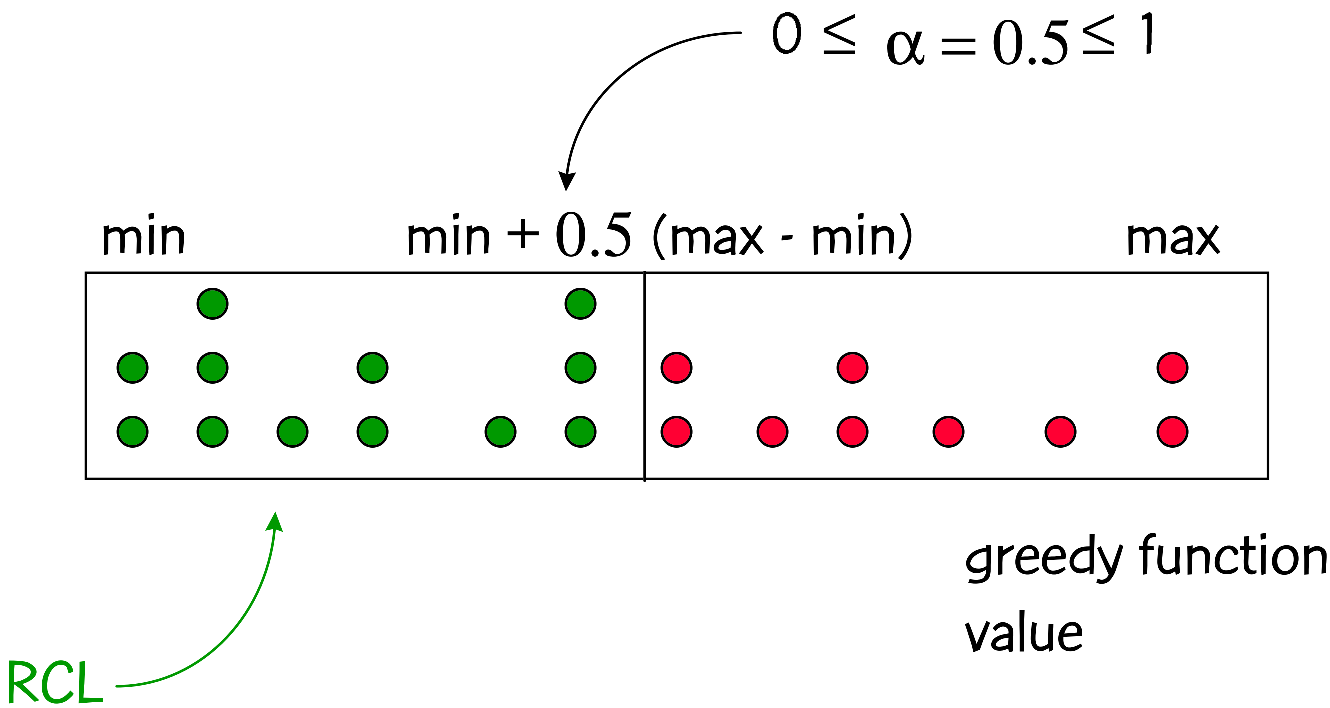


candidates = { 3 , 1 }

greedy (3) = 20; greedy (1) = 25

greedy choice = 3

Candidate list restriction parameter α



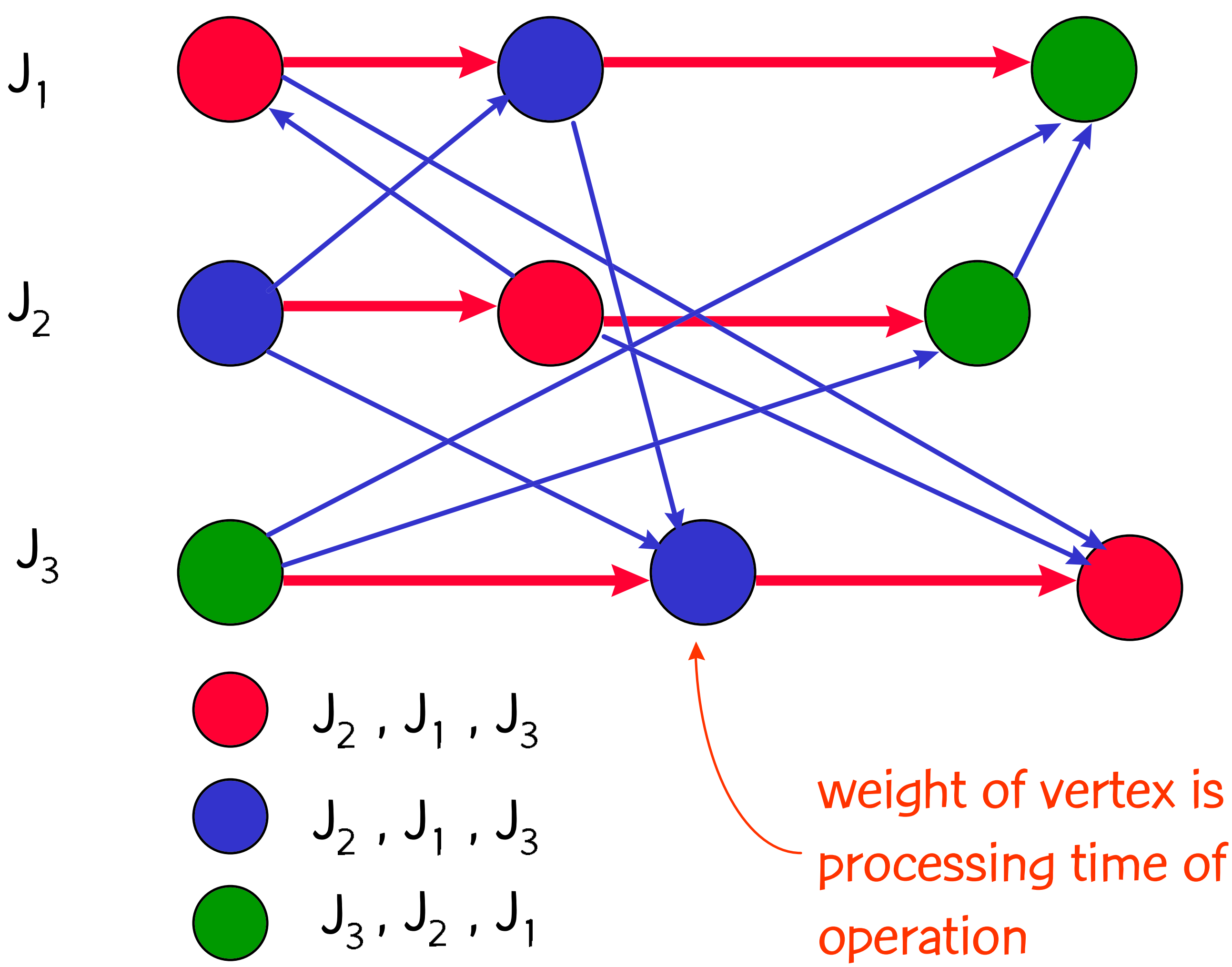
$$\alpha = 0.5$$

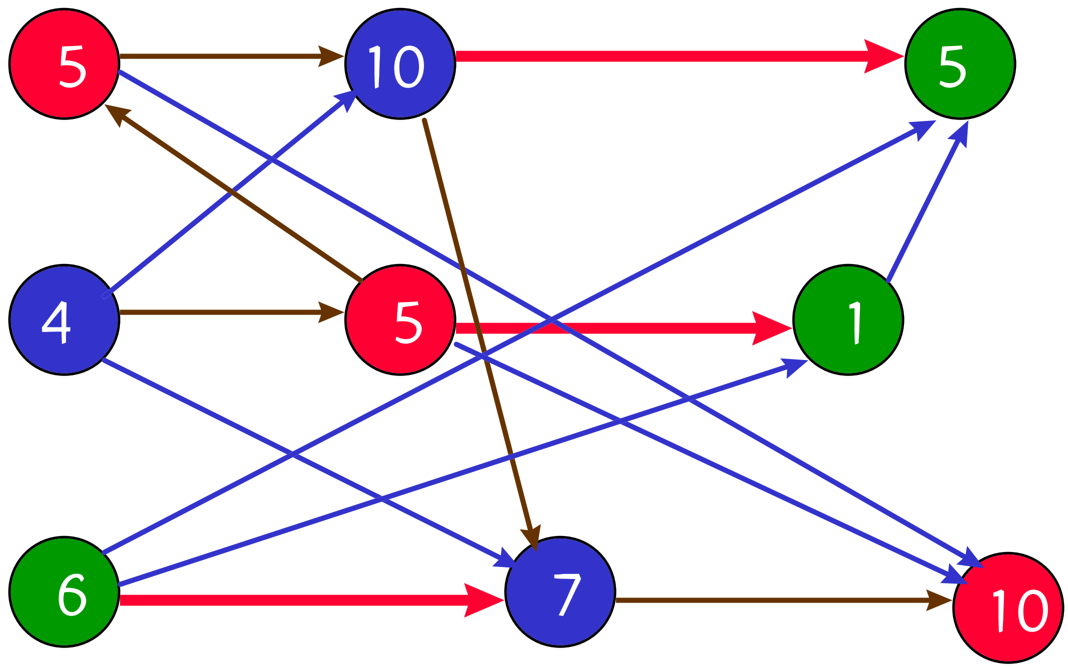
Local neighborhood structure

- We use standard disjunctive graph representation of job shop schedule [Roy & Sussmann, 1964]
- $G = (V, A, E)$, where
  - V is set of operations
  - A is set of arcs connected consecutive operations of the same job
  - E is set of edges connecting operations that must be executed on the same machine

Local neighborhood structure

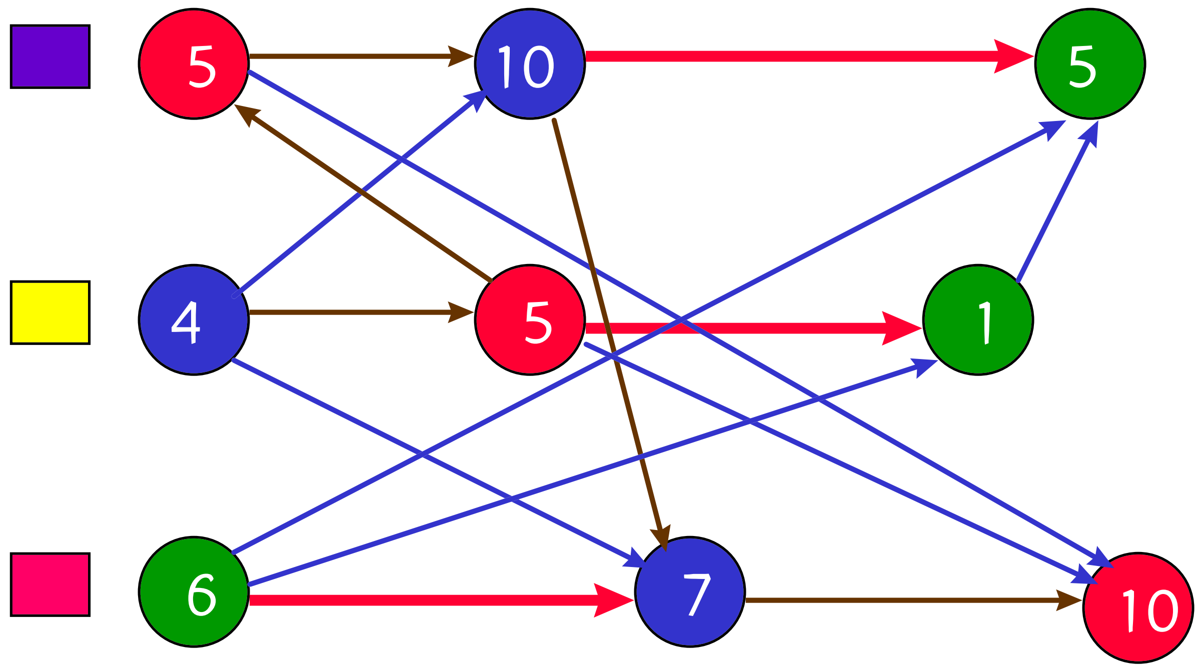
disjunctive graph representation of schedule





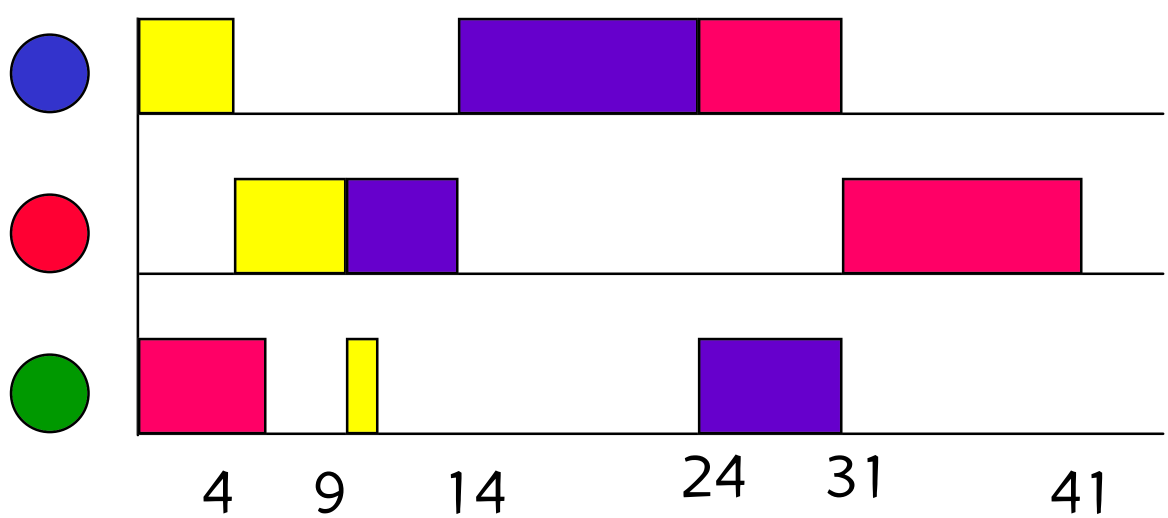
critical path: longest path in digraph

length of critical path = makespan = 41



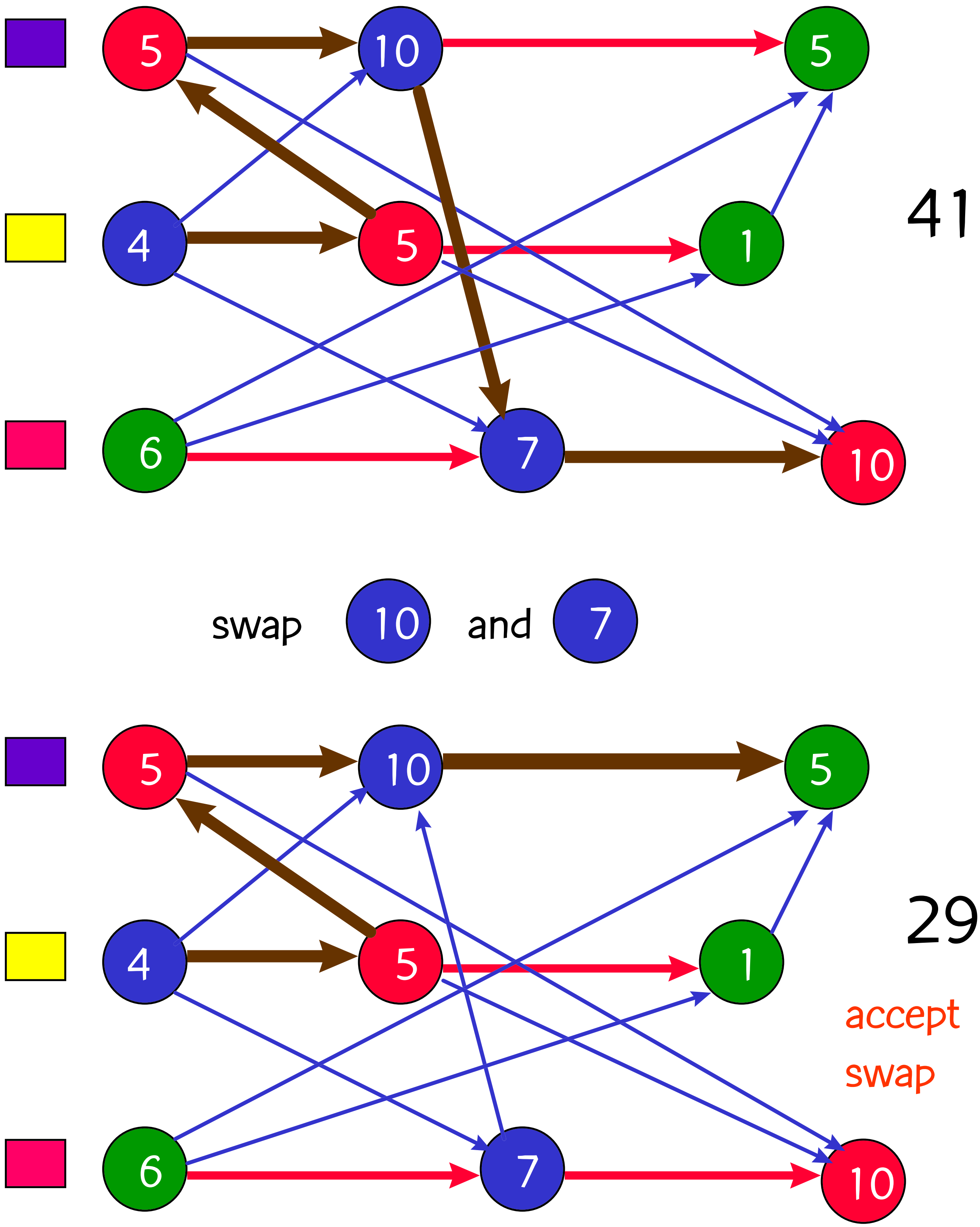
critical path: longest path in digraph

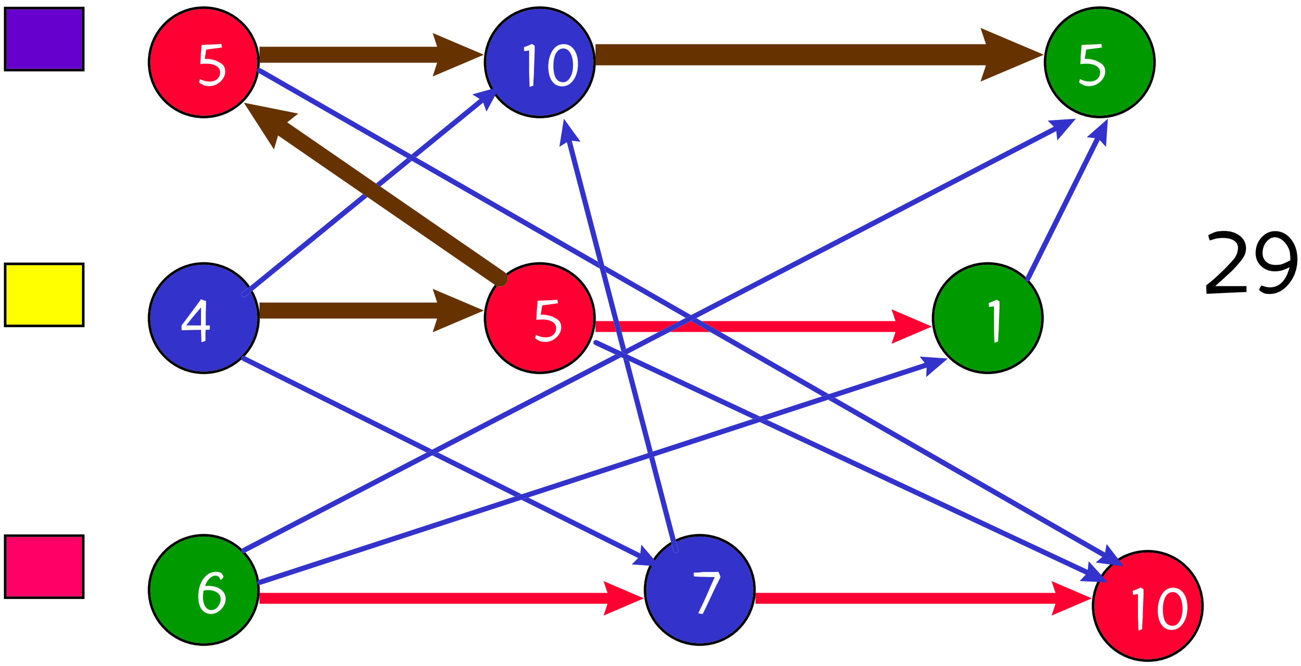
length of critical path = makespan = 41

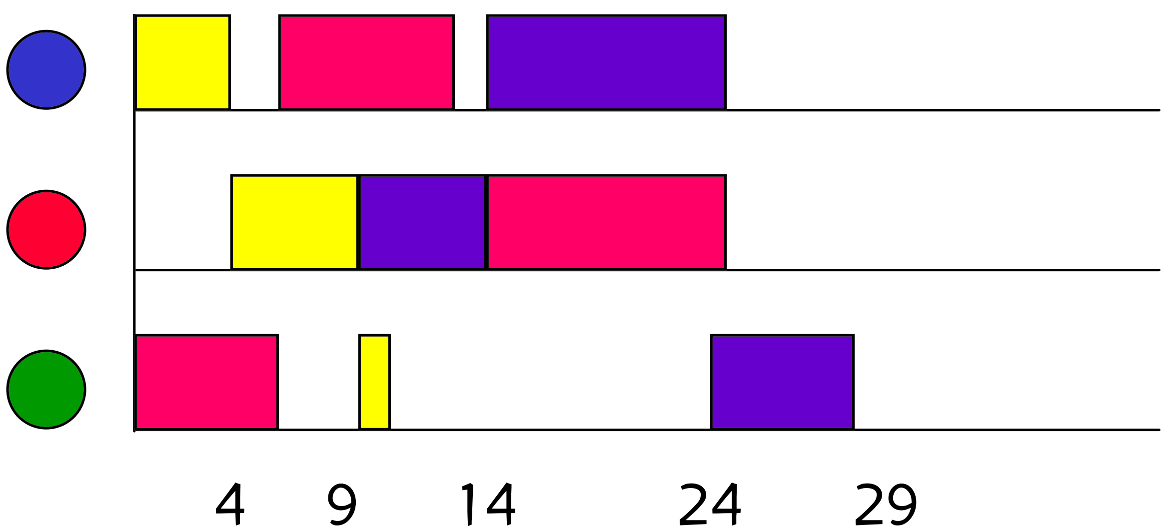


Local search

- For every pair of operations v and w, such that:
  - v and w are sucessive operations on a machine
  - (v, w) are is in the critical path
    - swap order of v and w, i.e. make arc (v,w) into (w,v)
    - find critical path
    - if critical path length is shorter, accept swap, else reject it


5
10
5
4
5
1
41
6
7
10
swap
10
and
7
5
10
5
4
5
1
29
accept
swap
6
7
10


5
10
5
4
5
1
29
6
7
10


4
9
14
24
29

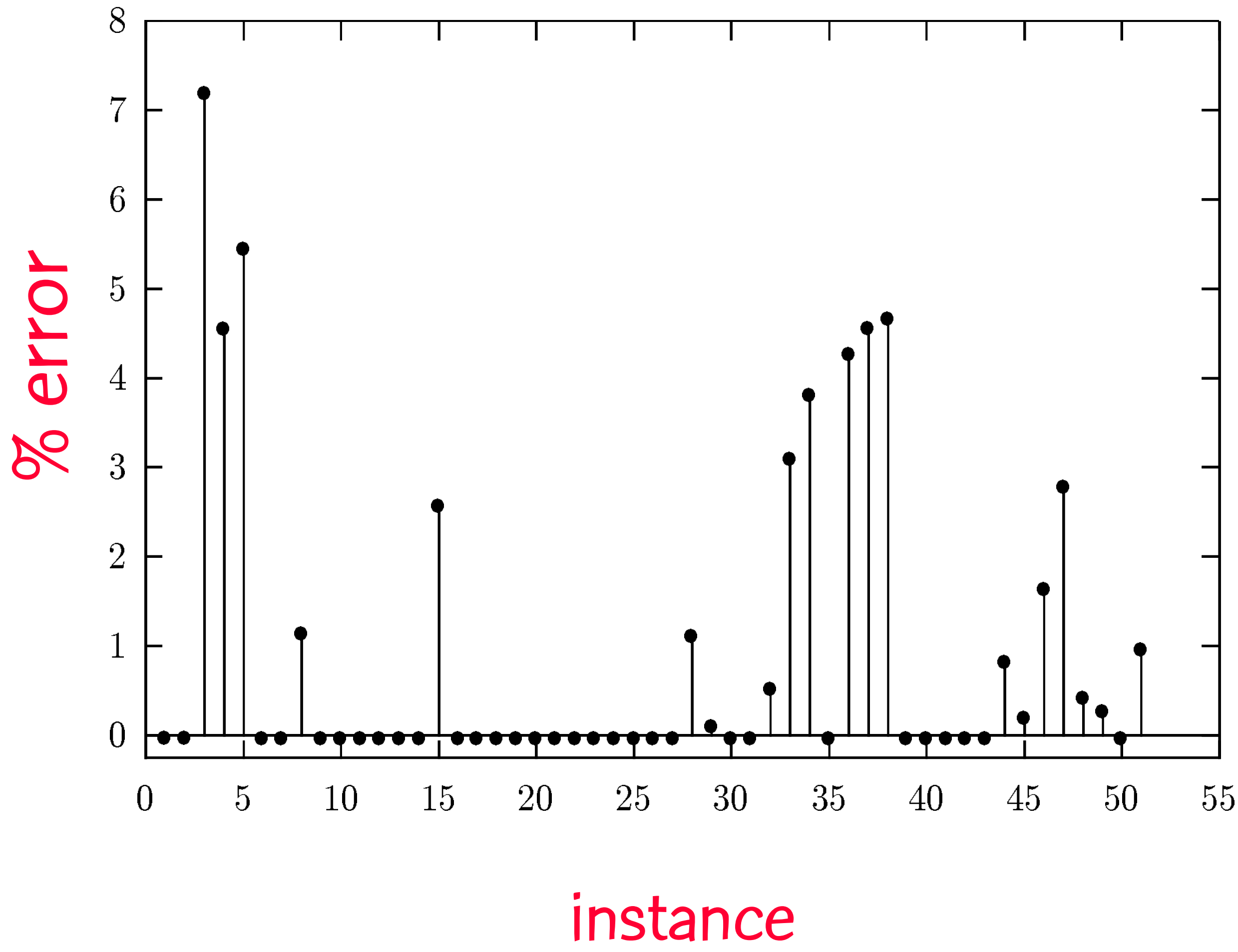
Recomputing critical path

- use variation of Bellman's labelling algorithm (Taillard, 1994)
- critical path can be recomputed in $O(N)$ operations, where
  - N is the number of operations

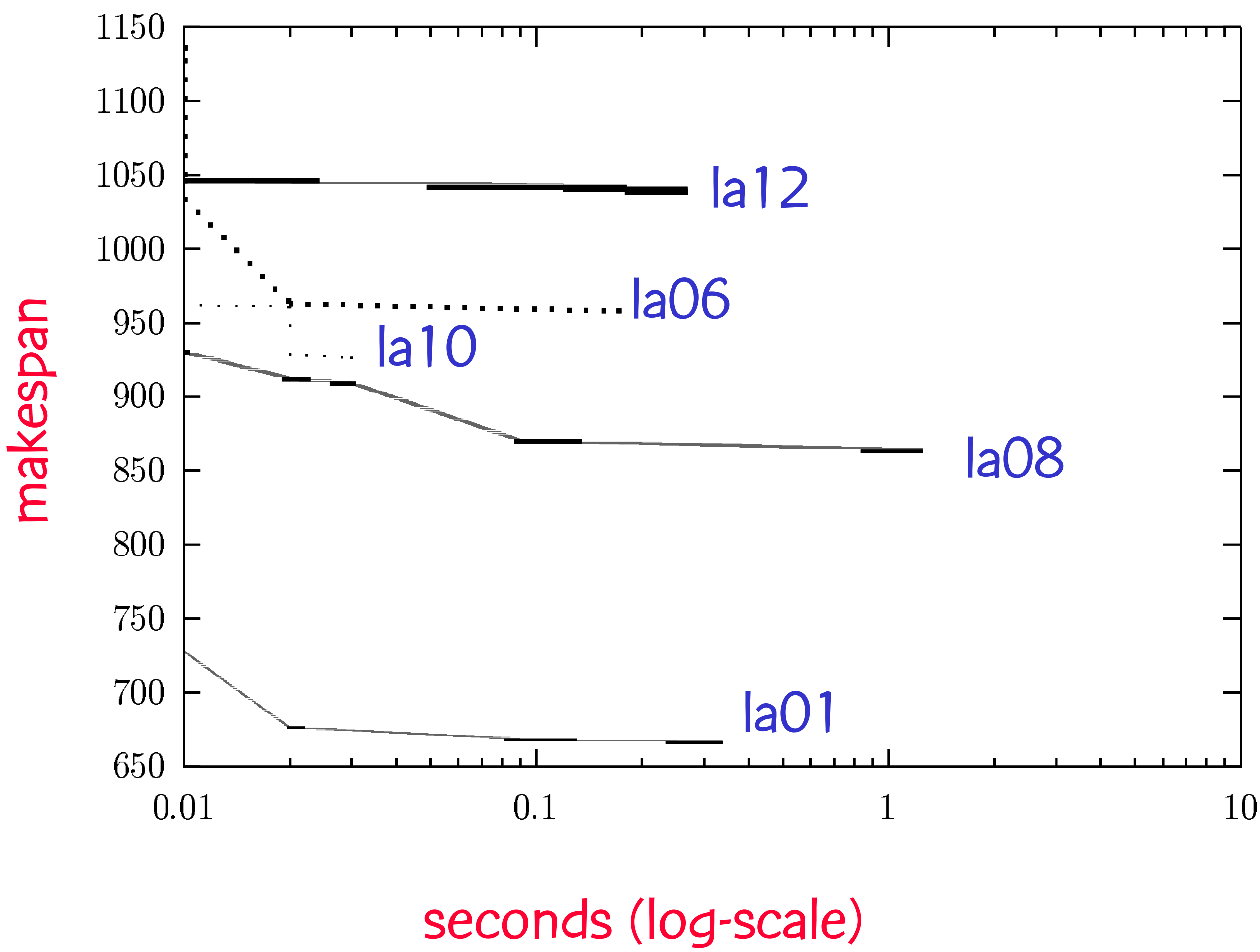
Computational experience

- Tested GRASP on large number of standard JSS test problems
- Ran GRASP
  - with RCL parameter $\alpha = 0.5$
  - for at most 10,000,000 iterations
  - on 10 SGI R10000 processors in parallel
- Stop when
  - max number of iterations is reached
  - best known solution is found

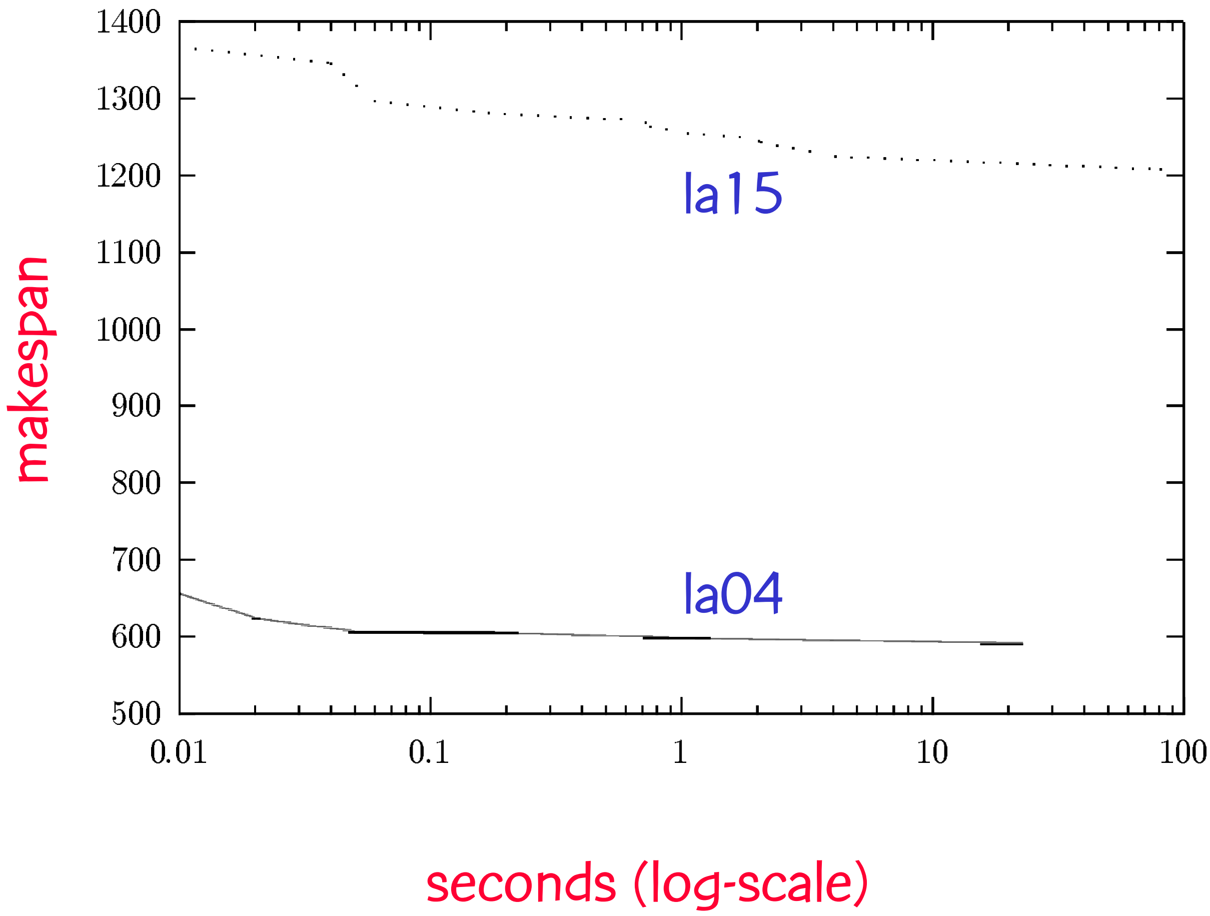
Percentage error on all instances



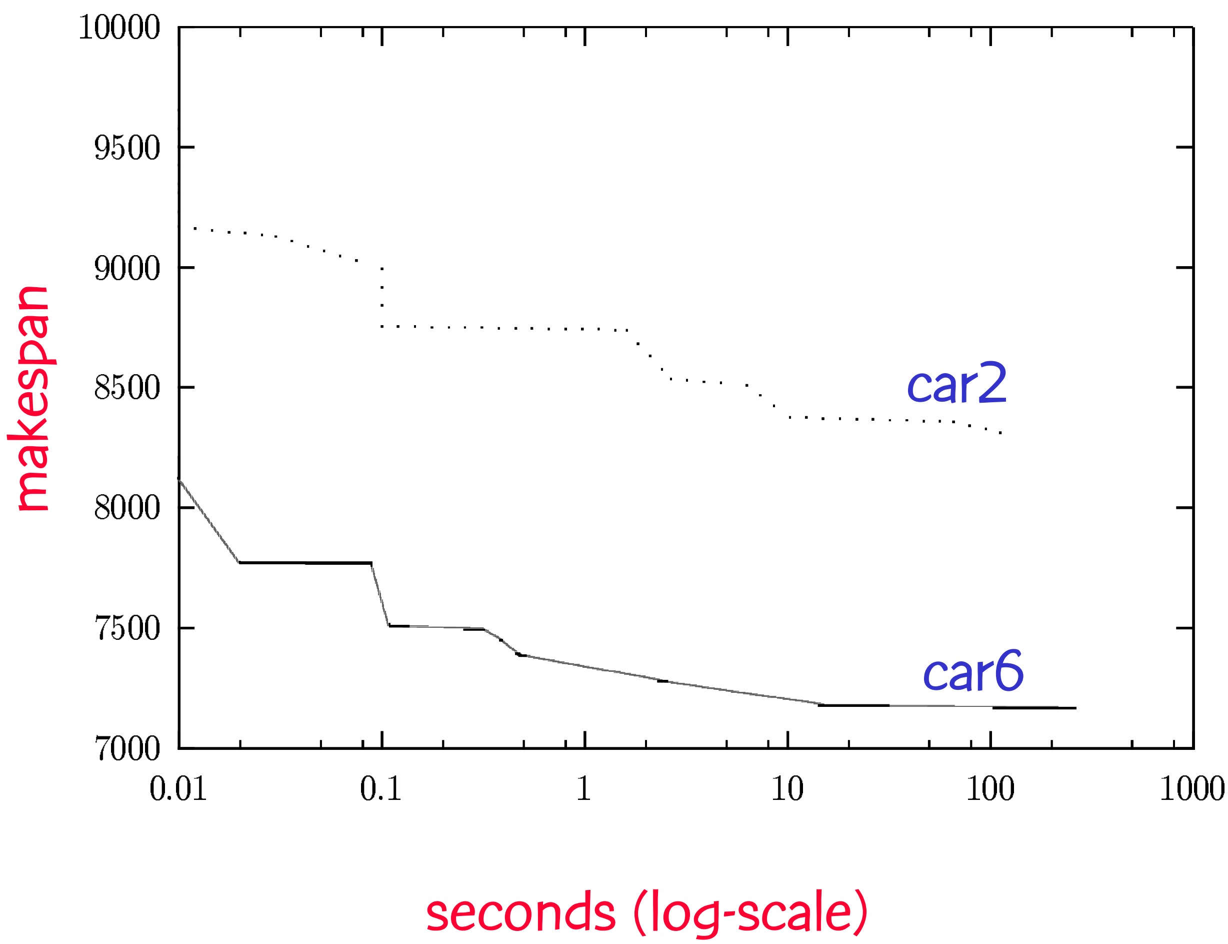
Best known solution



Best known solution



Best known solution



Best known solution