Case study: Time tabling

Rossi-Doria et al. 2002 `http://dx.doi.org/10.1007/978-3-540-45157-0_22`

- Set of events E, set of rooms R, set of students S, set of features F
- Each student attends a number of events and each room has a size.
- Assign all events a timeslot and a room so that the following *hard constraints* are satisfied:
  - no student attends more than one event at the same time.
  - the room is big enough for all attending students and satisfies all features required by the event.
  - only one event is in each room at any timeslot.

Case study: Time tabling (2)

- Penalties for *soft constraint* violations
  - a student has a class in the last slot of a day.
  - a student has more than two classes in a row.
  - a student has a single class on a day.
- Objective: Minimize number of soft constraint violations in a feasible solution

Common neighborhood structure

- *Solution* $\rightsquigarrow$ ordered list of length $|E|$
  The i-th element indicates the timeslot to which event i is assigned.
- Room assignments generated by matching algorithm.
- *Neighborhood:* $N = N_1 \cup N_2$
  - N_1 moves a single event to a different timeslot
  - N_2 swaps the timeslots of two events.

Common local search procedure

Stochastic first improvement local search

- Go through the list of all the events in a random order.
- Try all the possible moves in the neighbourhood for every event involved in constraint violations, until improvement is found.
- Solve hard constraint violations first.
  If feasibility is reached, look at soft constraint violations as well.

Metaheuristics

1. Evolutionary algorithm
2. Ant colony optimization
3. Iterated local search

4. Simulated annealing
5. Tabu search

1. Evolutionary algorithm

- *Steady-state evolution process:* at each generation only one couple of parent individuals is selected for reproduction.
- *Tournament selection:* choose randomly a number of individuals from the current population and select the best ones in terms of fitness function as parents.
- *Fitness function:* Weighted sum of hard and soft constraint violations,

$$f(s) := \#hcv(s) \cdot C + \#scv(s)$$

1. Evolutionary algorithm (2)

- *Uniform crossover:* for each event a timeslot's assignment is inherited from the first or second parent with equal probability.
- *Mutation:* Random move in an extended neighbourhood (3-cycle permutation).
- *Search parameters:* Population size $n = 10$, tournament size = 5, crossover rate $\alpha = 0.8$, mutation rate $\beta = 0.5$
- Find a balance between the number of steps in local search and the number of generations.

2. Ant colony optimization

- At each iteration, *each of m ants constructs,* event by event, a *complete assignment* of the events to the timeslots.
- To make an assignment, an ant takes the next event from a pre-ordered list, and probabilistically chooses a timeslot, guided by two types of information:
  1. *heuristic information:* evaluation of the constraint violations caused by making the assignment, given the assignments already made,
  2. *pheromone information:* estimate of the utility of making the assignment, as judged by previous iterations of the algorithm.
- *Matrix* of pheromone values $\tau : E \times T \to \mathbb{R}_{\geq 0}$.
  Initialization to a parameter τ_0, update by local and global rules.

2. Ant colony optimization (2)

- An event-timeslot pair which has been part of good solutions will have a high pheromone value, and consequently have a higher chance of being chosen again.
- At the end of the iterative construction, an event-timeslot assignment is converted into a candidate solution (timetable) using the matching algorithm.
- This candidate solution is further improved by the local search routine.
- After all m ants have generated their candidate solution, a global update on the pheromone values is performed using the best solution found since the beginning.

3. Iterated local search

- Provide new starting solutions obtained from **perturbations** of a current solution
- Often leads to far better results than using random restart.
- Four subprocedures
  1. *GenerateInitialSolution:* generates an initial solution s_0
  2. *Perturbation:* modifies the current solution s leading to some intermediate solution s',
  3. *LocalSearch:* obtains an improved solution s'',
  4. *AcceptanceCriterion:* decides to which solution the next perturbation is applied.

Perturbation

- *Three types of moves*

  P1: choose a different timeslot for a randomly chosen event;

  P2: swap the timeslots of two randomly chosen events;

  P3: choose randomly between the two previous types of moves and a 3-exchange move of timeslots of three randomly chosen events.

- *Strategy*
  - Apply each of these different moves k times, where k is chosen of the set $\{1; 5; 10; 25; 50; 100\}$.
  - Take random choices according to a uniform distribution.

Acceptance criteria

- Random walk: Always accept solution returned by local search
- Accept if better
- Simulated annealing

  SA1: $P_1(s, s') = e^{-\frac{f(s)-f(s')}{T}}$

  SA2: $P_2(s, s') = e^{-\frac{f(s)-f(s')}{T \cdot f(s_{best})}}$

Best parameter setting (for medium instances):

P1, $k = 5$, **SA1** with $T = 0.1$

4. Simulated annealing

Two phases

1. Search for feasible solutions, i.e., satisfy all hard constraints.
2. Minimize soft constraint violations.

Strategies

- *Initial temperature:* Sample the neighbourhood of a randomly generated solution, compute average value of the variation in the evaluation function, and multiply this value by a given factor.

- *Cooling schedule*
  1. Geometric cooling: $T_{n+1} = \alpha \times T_n, \quad 0 < \alpha < 1$
  2. Temperature reheating: Increase temperature if *rejection ratio* (= number of moves rejected/number of moves tested) exceeds a given limit.
- *Temperature length* (number of iterations at each temperature): Proportional to the size of the neighborhood

5. Tabu search

- Moves done by moving one event or by swapping two events.
- Explore solutions that do not decrease the objective function value
- *Tabu list:* Forbid a move if at least one of the events involved has been moved less than l steps before.
- *Size of tabu list l:* number of events divided by a suitable constant k (here $k = 100$).
- *Variable neighbourhood set:* every move is a neighbour with probability 0.1 $\rightsquigarrow$ decrease probability of generating cycles and reduce the size of neighbourhood for faster exploration.
- *Aspiration criterion:* perform a tabu move if it improves the best known solution.

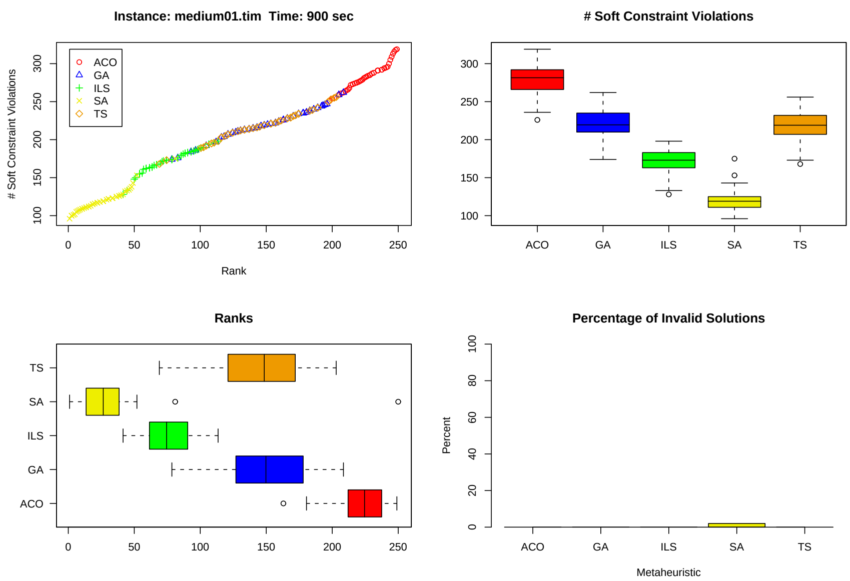
Evaluation

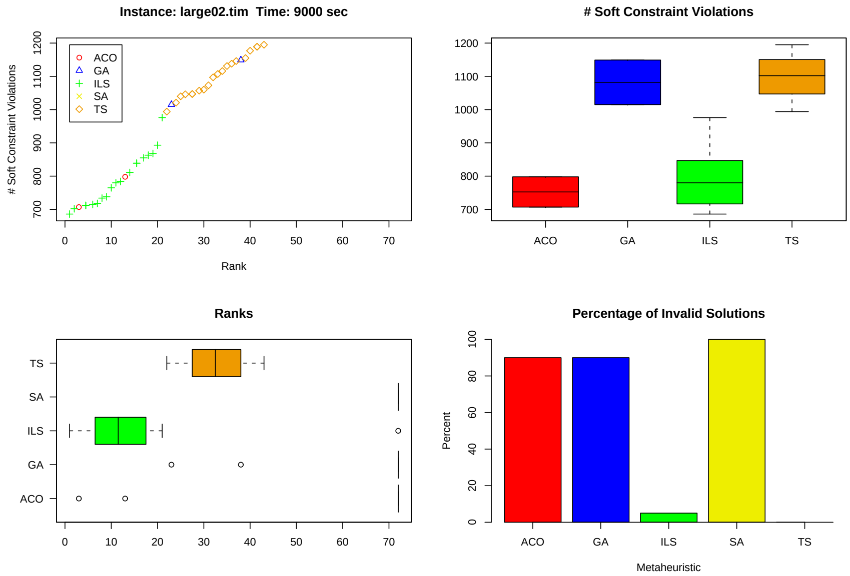
`http://iridia.ulb.ac.be/~msampels/ttmn.data/`

- 5 small, 5 medium, 2 large instances

Type	small	medium	large
$\|E\|$	100	400	400
$\|S\|$	80	200	400
$\|R\|$	5	10	10

- 500 resp. 50 resp. 20 independent trials per metaheuristic per instance.
- Diagrams show results of all trials on a single instance.
- Boxes show the range between 25% and 75% quantile.





Evaluation (2)

- *Small:* All algorithms reach feasibility in every run,
  ILS best, TS worst overall performance
- *Medium:* SA best, but does not achieve feasibility in some runs. ACO worst.
- *Large01:* Most metaheuristics do not even achieve feasibility. TS feasibility in about 8% of the trials.
- *Large02:* ILS best, feasibility in about 97% of the trials, against 10% for ACO and GA. SA never reaches feasibility. TS gives always feasible solutions, but with worse results than ILS and AC0 in terms of soft constraints.