A.A. 2016/2017

Optimization methods and algorithms

ColoTe Project

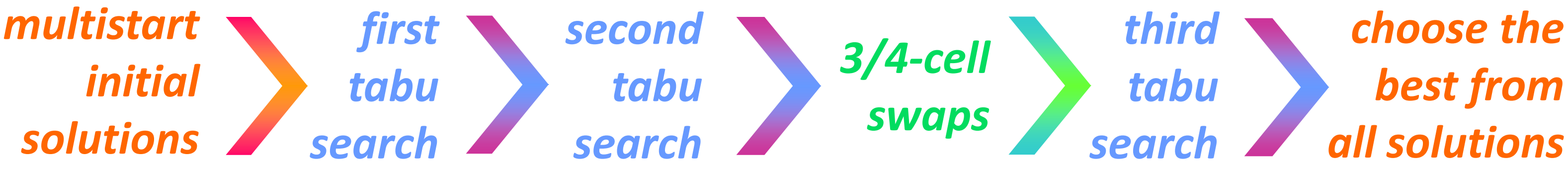
a multistart tabu search approach

group **26**

235509 CLAUDIO SCALZO
204318 THAI HAO MARCO VAN
234391 GABRIELE MILAURO

STEFANO MONTI 235409
MARCO SCHIUMA 235860
GIANFRANCO FANTAPPIE 233537

Algorithm overview



multistart initial solutions

first tabu search

second tabu search

3/4-cell swaps

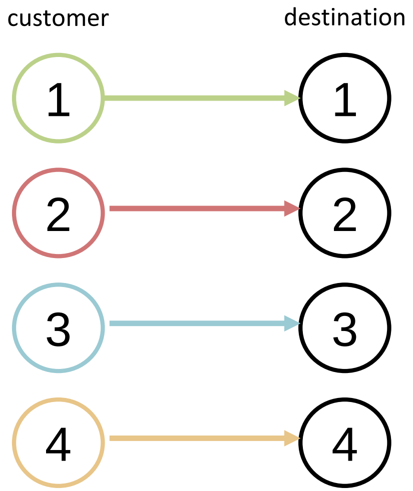
third tabu search

choose the best from all solutions

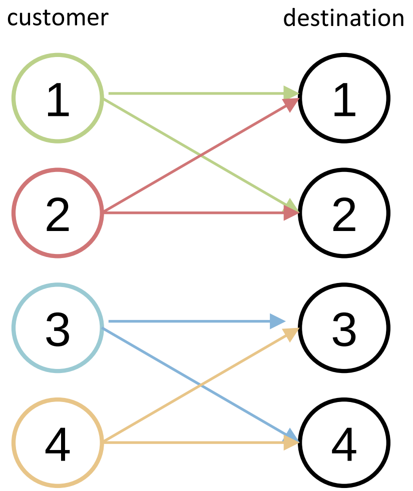
- Each **initial solution** is created by a thread with a different parameter
  - This variation produces **different initial solutions** (and, as a consequence, different final solutions)
  - Initial solutions make use of **arches sorting** (according to their costs) to obtain a good initial gap
- **Tabu search** performs swaps of users among cells. The program uses three kinds of tabu search, each one distinct from the others because of the different swap used
- **Cell swaps** perform some particular moves not done by the tabu search. They work with already taken users and deal with three or four cell at a time
- The solution of each instance is **the best** of all the different solutions produced by each thread

Initial greedy solution

- Distributes customers to destinations in **different orders**
  - From the first to the last destination
  - From the last to the first destination
  - Pseudorandom way (using fixed *seeds*)
- Assigns to the destination the **cheapest** customers available
- **Variety coefficient** is used to diversify the solution created
- Tracks all the tasks done in **surplus**
- Tracks all the **customers** assigned to more than one destination



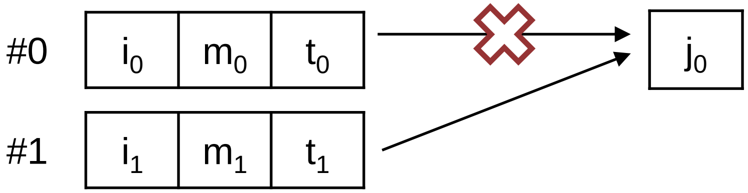
variety coefficient = 1



variety coefficient = 2

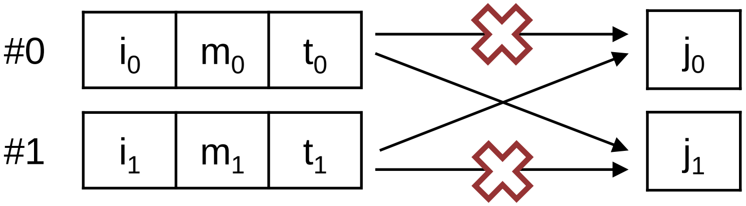
First tabu search: «free»

- **Neighborhood**: customers *not assigned* and close to the destination
- **Metaheuristic tuning:**
    - **Aspiration criterion**: best objective function
    - **Tabu move**: move of a customer to destination
    - **Reactive tabu list:**
        - Initial tenure: *10*
        - Tenure *+50%* if unimproving moves ≥ *4*
        - Tenure *-50%* if improving moves ≥ *16*
    - **Stopping criteria**: *55* number of iterations or *15* unimproving moves
    - **Neighborhood size**: *20* customers
- **Purpose**: improve the solution by swapping not assigned customers



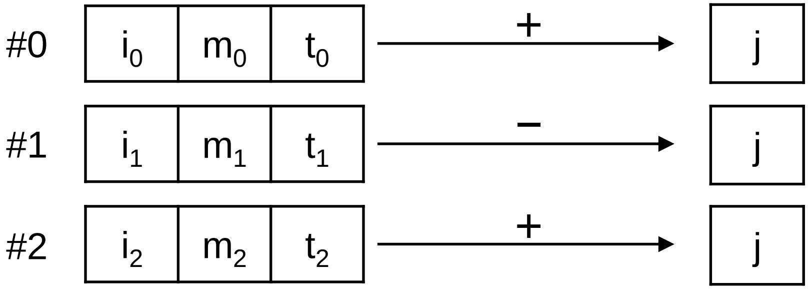
Second tabu search: «taken»

- **Neighborhood**: customers *already assigned* and close to the destination
- **Metaheuristic tuning:**
  - **Aspiration criterion**: best objective function
  - **Tabu move**: move of a customer to destination
  - **Reactive tabu list:**
    - Initial tenure: *10*
    - Tenure *+50%* if unimproving moves ≥ *4*
    - Tenure *-50%* if improving moves ≥ *16*
  - **Stopping criteria**: *40* number of iterations or *15* unimproving moves
  - **Neighborhood size**: *20* customers
- **Purpose**: improve the solution by swapping already assigned customers



First swap: «three-cell»

- Swap between 3 customers who have the same destination



#0	i_0	m_0	t_0	+	j
#1	i_1	m_1	t_1	−	j
#2	i_2	m_2	t_2	+	j

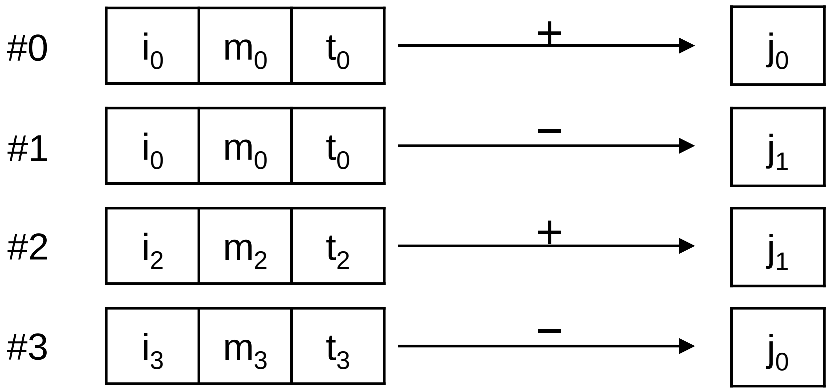
Neighborhood size = 20

Constraint: the tasks done by user *#1* must be replaced by the same amount of tasks, done by user *#0* and *#2*.

tasks m_1 = *tasks* m_0 + *tasks* m_2

Second swap: «four-cell»

- Swap between 4 customers who have two destinations in common



#0 i_0 m_0 t_0 $\xrightarrow{+}$ j_0

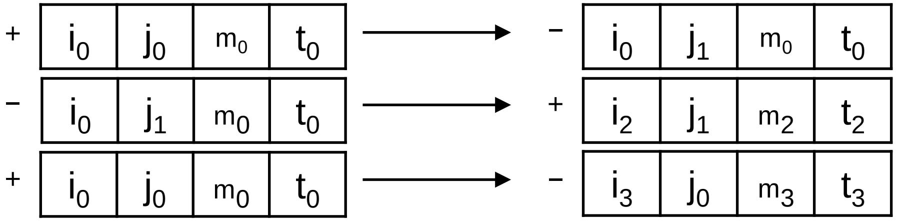
#1 i_0 m_0 t_0 $\xrightarrow{-}$ j_1

#2 i_2 m_2 t_2 $\xrightarrow{+}$ j_1

#3 i_3 m_3 t_3 $\xrightarrow{-}$ j_0

Neighborhood size = 20

Constraints:



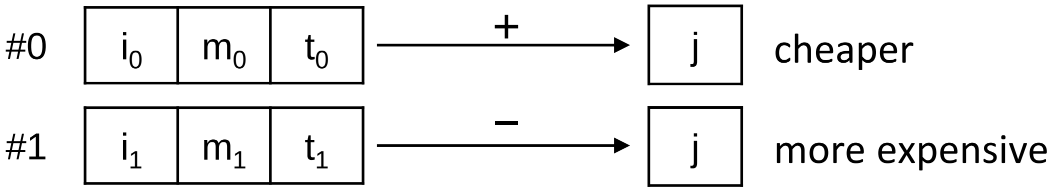
$+$ i_0 j_0 m_0 t_0 $\longrightarrow$ $-$ i_0 j_1 m_0 t_0

$-$ i_0 j_1 m_0 t_0 $\longrightarrow$ $+$ i_2 j_1 m_2 t_2

$+$ i_0 j_0 m_0 t_0 $\longrightarrow$ $-$ i_3 j_0 m_3 t_3

Third tabu search: «trim»

- **Neighborhood**: customers not assigned or assigned close to the destination
- **Metaheuristic tuning:**
    - **Aspiration criterion**: best objective function
    - **Tabu move**: move of a customer to destination
    - **Reactive tabu list:**
        - Initial tenure: *10*
        - Tenure *+50%* if unimproving moves ≥ *4*
        - Tenure *-50%* if improving moves ≥ *16*
    - **Stopping criteria**: *30* number of iterations or *5* unimproving moves
    - **Neighborhood size**: *10* customers
- **Purpose**:
    - remove the tasks in surplus
    - assign as much tasks as possible to the cheaper customers



#0 i_0 m_0 t_0 $\xrightarrow{+}$ j cheaper

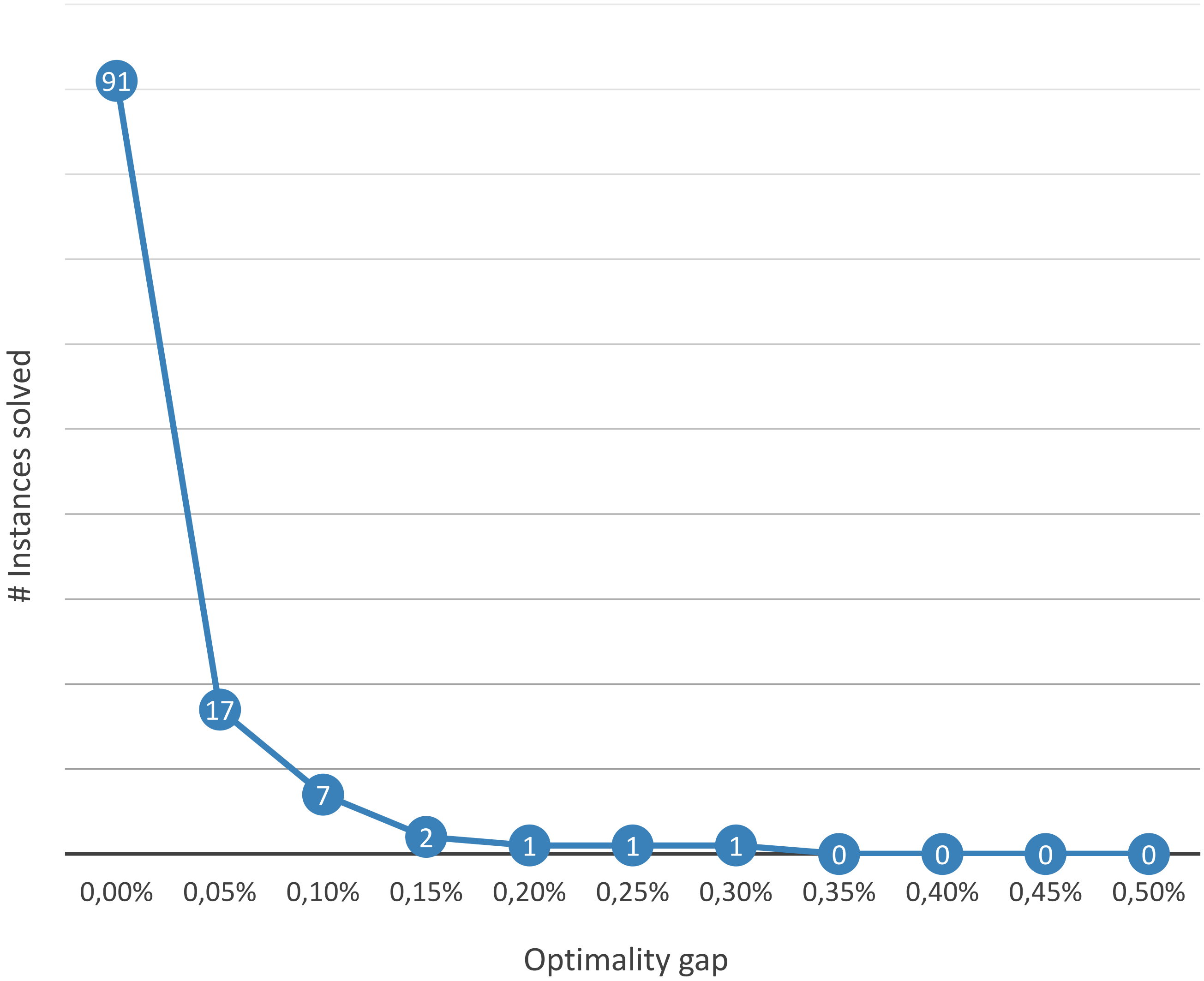
#1 i_1 m_1 t_1 $\xrightarrow{-}$ j more expensive

Overall results

INSTANCE TYPE	AVG TIME (i5 *@2.7GHz*)	AVG TIME (i7 *@2.6GHz*)	AVG OPTIMALITY GAP
30_1	0,348 s	0,345 s	**0,01 %**
30_20	0,188 s	0,151 s	**0,00 %**
100_1	2,491 s	1,590 s	**0,05 %**
100_20	0,880 s	0,732 s	**0,00 %**
300_20	2,348 s	2,525 s	**0,01 %**

- The program finds an **optimal solution** for more than the **75%** of the instances
- **Every** (non-optimal) solution has an optimality gap below 0.3%

Gaps frequency distribution



Overall results (hard instances)

- The program is also capable of solving the **hard** instances with very good gaps (always below 2%)

HARD INSTANCE	TIME	OPTIMALITY GAP
30_1_ST_0	0,246 s	**0,00 %**
30_1_TL_0	0,325 s	**0,07 %**
30_1_TT_0	0,309 s	**1,72 %**
30_20_ST_0	0,438 s	**1,89 %**
30_20_TL_0	0,166 s	**0,40 %**
30_20_TT_0	0,201 s	**1,49 %**
100_1_ST_0	0,861 s	**1,93 %**
100_1_TL_0	1,971 s	**0,38 %**
100_1_TT_0	1,646 s	**0,42 %**
100_20_TL_0	0,684 s	**0,12 %**
100_20_TT_0	1,050 s	**0,75 %**

Thanks for your attention

group **26**