A Kernel Search Approach for the Time-Dependent Rural Postman Problem

R. Mansini[2], G. Ghiani[1], E. Guerriero[1], R. Zanotti[2]

1 Dept. of Engineering for Innovation, University of Salento, Italy
2 Dept. of Information Engineering, University of Brescia, Italy

ODS 2019, Genova 4-7, September 2019

Outline

- The Problem
  - The Rural Postman Problem (RPP)
  - The Time-Dependent RPP (TDRPP)
  - Motivation
  - Literature
- Arc-Path integer formulation
  - Pros and Cons
- Kernel Search (KS)
  - Main Idea
  - Basic KS
  - KS with ARC-PATH
  - KS with B&B
- Computational results
- Conclusions

The Rural Postman Problem (RPP)

Given:

- $G = (V, A)$ directed connected graph
- a set of nodes $V = N \cup \{0\}$, where 0 is the depot
- a set of arcs $A = \{(i, j) : i, j \in V\}$
- a set $R \subseteq A$ of required arcs
- a nonnegative travel distance L_{ij} for each arc $(i, j) \in A$

Objective: Find a minimum-cost tour traversing each required arc at least once.

Time-dependent RPP (TDRPP)

Traveling time becomes a function of the time t an arc (i, j) starts to be traversed:

- $\tau_{ij}(t)=$ travel time function, depending on time t, associated with each arc $(i, j) \in A$
- First-in First-out (FIFO) property has to be satisfied: a later start time cannot result in an earlier arrival time

Objective: Find a least duration tour traversing each requested arc at least once, with the vehicle leaving depot at time 0.

Motivation

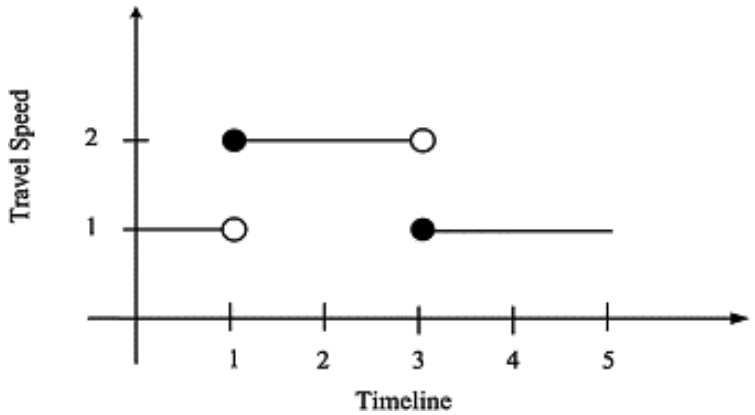
Applications of the TDRPP:

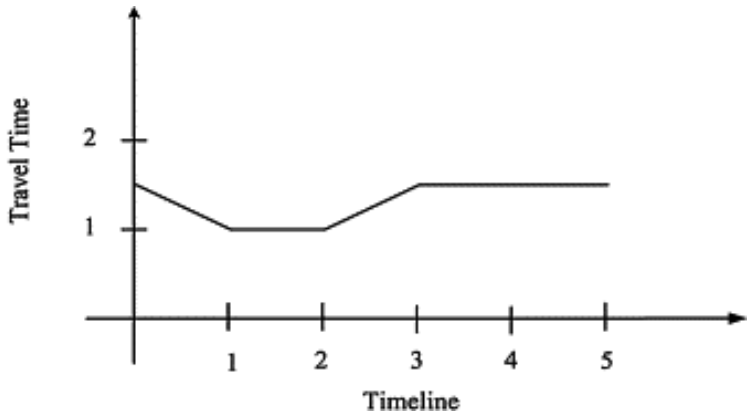
1. Scheduling with time dependent processing times
2. Service Route planning in road networks (garbage collection, streets cleaning)
3. Robot motion planning in presence of moving obstacle

Potential causes of variability in travel times:

1. hourly (daily, weekly or seasonal) traffic congestion
2. random events (accidents, weather conditions)

Some remarks on time-dependent





- ✓ to guarantee FIFO property $\tau_{ij}(t)$ cannot be a discrete function of $t \rightarrow$ use of speed functions
- ✓ If all the arcs share the same traffic pattern the TDRPP can be solved as its time-independent counterpart with suitably-defined (constant) travel times (Cordeau et al. [2014])

Some Assumptions

Given a time horizon $[0, T]$, within which a tour has to be completed, we assume:

1. T is partitioned into H subintervals $[T_h, T_{h+1}]$, $h = 0, ..., H-1$, where $T_0 = 0$ and $T_H = T$.
2. $\tau_{ij}(t)$ are continuous piecewise linear functions and satisfy the first-in-first-out (FIFO) property
3. $\tau_{ij}(t)$ can be generated from the model proposed by Ichoua et al. (2003) with constant stepwise speed functions, where $v_{ij}(t) = v_{ijh}$ with $t \in [T_h, T_{h+1}]$ and $h = 0, ..., H-1$

Notation of the TDRPP 1/2

$$v_{ijh} = u_{ij} b_h \delta_{ijh} \quad (i,j) \in A, h = 0, ..., H-1$$

- u_{ij} is the maximum travel speed across arc $(i,j) \in A$ during $[0,T]$
- $b_h \in [0,1]$ is the best (lightest) congestion factor in $[T_h, T_{h+1}]$
- $\delta_{ijh} \in [0,1]$ is the degradation of the congestion factor of arc (i,j) in $[T_h, T_{h+1}]$ with respect to the less congested arc in $[T_h, T_{h+1}]$

Common congestion

If

$$\min_{\substack{(i,j)\in A \\ h=0,\ldots,H-1}} \delta_{ijh} = 1$$

all $(i,j) \in A$ share a common congestion factor b_h in $[T_h, T_{h+1}]$.

Notation of the TDRPP 2/2

Given a start time t and a route p $(s = v_1, v_2, ..., v_k = d)$ from a node $s \in V$ to a node $d \in V$ (possibly coincident with s), its traversal time $z_p(t)$ is defined recursively as:

$$z_{(v_1,...,v_i)}(t) = z_{(v_1,...,v_{i-1})}(t) + \tau_{v_{i-1},v_i}(t + z_{(v_1,...,v_{i-1})}(t))$$

with the initialization $z_{(v_1,v_2)}(t) = \tau_{v_1,v_2}(t)$

Literature

Literature on ARPs with time dependencies

- CARP time-dependent service costs (Tagmouti et al. [2007])
- Time-dependent RPP:
  - Tan and Sun [2011]: ARC-PATH formulation
  - Calogiuri et al. [2019]: Branch-and-Bound

Literature on Kernel Search (KS)

- Angelelli, Mansini, Speranza [2010]: Multi-dimensional KP
- Angelelli, Mansini, Speranza [2012]: Portfolio Selection Problem
- Guastaroba, Savelsbergh, Speranza [2017]: MIP problems
- Hanafi, Mansini, Zanotti [2018]: Team Orienteering variant

Arc-Path formulation (Tan and Sun [2011]) 1/3

Main idea: a solution (tour) is formulated as an alternation of required arcs and paths:

$$(0 - \underbrace{p_1(\tau_1)}_{\text{first path}} - \underbrace{a_1(t_1)}_{\text{first arc}} - p_2(\tau_2) - \ldots - a_K(t_K) - p_{K+1}(\tau_{K+1}) - 0)$$

Problem formulation: four main sets of binary variables

- x_{ij}^k= 1 if (i, j) is the k-th required arc of the tour; 0 otherwise.
- y_{ij}^k= 1 if (i, j) is in the k-th path of the tour; 0 otherwise.
- $\delta_{ij}^{k,h}$= 1 if (i, j) is served as the k-th required arc of the tour in interval h-th; 0 otherwise.
- $\gamma_{ij}^{k,h}$= 1 if (i, j) is traversed in the k-th path of the tour in interval h-th; 0 otherwise.

Arc-Path formulation (Tan and Sun [2011]) 2/3

$$min \sum_{k=1}^{K} \sum_{h=1}^{H} \sum_{(i,j)\in R} D_{ij}^{h} \delta_{ij}^{k,h} + \sum_{k=1}^{K+1} \sum_{h=1}^{H} \sum_{(i,j)\in R} D_{ij}^{h} \gamma_{ij}^{k,h} \tag{1}$$

$$\sum_{i=1}^{K} x_{ij}^{k} = 1 \quad (i,j) \in R \tag{2}$$

$$\sum_{(i,j)\in R} x_{ij}^{k} = 1 \quad k = 1, ..., K \tag{3}$$

$$\sum_{j\in N(i)^+} y_{ij}^{k} \leq 1 \quad i \in V, k = 1, ..., K \tag{4}$$

$$\sum_{j\in N(i)^-} y_{ji}^{k} \leq 1 \quad i \in V, k = 1, ..., K+1 \tag{5}$$

$$\sum_{j\in N(i)^+} y_{ij}^{k} - \sum_{j\in N(i)^-} y_{ji}^{k} = \sum_{j\in N(i)^-} x_{ji}^{k-1} - \sum_{j\in N(i)^+} x_{ij}^{k} \quad i \in V, k = 2, ..., K \tag{6}$$

$$\sum_{j\in N(i)^+} y_{ij}^{1} - \sum_{j\in N(i)^-} y_{ji}^{1} = \begin{cases} 1 - \sum_{j\in N(i)^+} x_{ij}^{1} & i = 0 \\ -\sum_{j\in N(i)^+} x_{ij}^{1} & i \neq 0 \end{cases} \tag{7}$$

Arc-Path integer formulation 3/3

$$\sum_{j \in N(i)^+} y_{ij}^{K+1} - \sum_{j \in N(i)^-} y_{ji}^{K+1} = \begin{cases} \sum_{j \in N(i)^-} x_{ji}^K - 1 & i = 0 \\ \sum_{j \in N(i)^+} x_{ji}^K & i \neq 0 \end{cases} \tag{8}$$

$$\tau_0^1 \geq t_0 \tag{9}$$

$$t_i^k \geq \tau_i^k \quad i \in V, k = 1, 2, ..., K, \tag{10}$$

$$\tau_k^i \geq t_i^{k-1} \quad i \in V, k = 2, ..., K+1, \tag{11}$$

$$t_j^k - t_i^k \geq D_{ij}^h + \delta_{ij}^{k,h} B - B \quad k = 1, 2, ..., K, h = 1, ..., H, (i,j) \in R; \tag{12}$$

$$\tau_j^k - \tau_i^k \geq D_{ij}^h + \gamma_{ij}^{k,h} B - B \quad k = 1, 2, ..., K+1, h = 1, ..., H, (i,j) \in A; \tag{13}$$

$$\sum_{h=1} \delta_{ij}^{k,h} = x_{ij}^k \quad (i,j) \in R, k = 1, ..., K; \tag{14}$$

$$\sum_{h=1} \gamma_{ij}^{k,h} = y_{ij}^k \quad (i,j) \in A, k = 1, ..., K+1; \tag{15}$$

$$T_{ij}^{h-1} \delta_{ij}^{k,h} \leq t_i^k \leq T_{ij}^h - B(\delta_{ij}^{k,h} - 1) \ (i,j) \in E, k = 1, ..., K; h = 1, ..., H \tag{16}$$

$$T_{ij}^{h-1} \gamma_{ij}^{k,h} \leq \tau_i^k \leq T_{ij}^h - B(\gamma_{ij}^{k,h} - 1) \ (i,j) \in A, k = 1, ..., K+1; h = 1, ..., H \tag{17}$$

Pros and Cons of the ARC-PATH formulation

- PROS:
  - Provide a mathematical formulation that can be exploited to get information on the problem: reduced cost values, LP solution values, Driebeek's Up-Down penalties
- CONS:
  - The formulation requires a huge number of variables and constraints
  - The formulation is based on stepwise travel time functions (NO FIFO property)
  - Solving LP relaxation of mid-size instances requires a huge amount of time

Main Idea to apply Kernel Search to TDRPP

Provide a new variant of the Kernel Search that:

1. exploits the information provided by the ARC-PATH model to sort items and construct buckets

$$\Downarrow$$

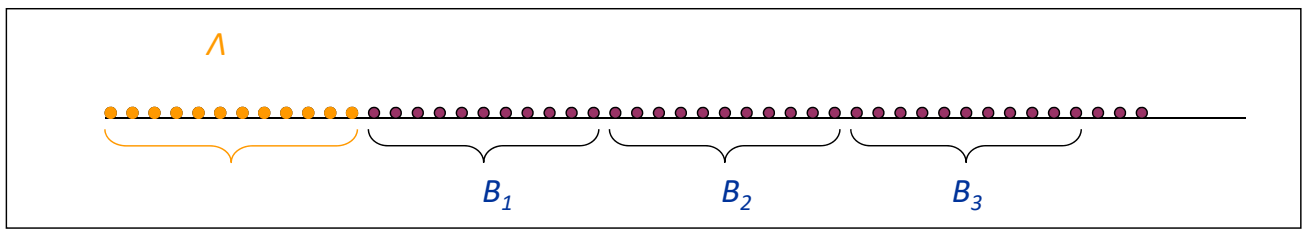
KS is constructed using as ITEM the combination

$$(\underbrace{\text{REQUIRED ARC}}_{a \in R}, \underbrace{\text{POSITION IN THE PATH}}_{k}, \underbrace{\text{INTERVAL}}_{h})$$

2. uses the ARC-PATH model to formulate subproblems
3. MIP Solver: Gurobi 8.1.0

Kernel Search: Main features

1. Sorting of items (variables) according to a predefined criterion (variable LP values, reduced costs)
2. Identification of a Kernel Set (Λ) of promising variables (more probably selected in an optimal solution) and partition of remaining variables into buckets



3. Construction and solution of a sequence of small subproblems by a MIP solver:

$$\text{Subproblem} = \text{MIP}(\Lambda \cup B) \qquad B \in \mathcal{B}$$

Kernel Search: Critical Aspects

1. Manage trade-off between solution quality and computational effort (appropriate size of Kernel Set and buckets $\rightarrow$ size of subproblems)

Kernel Search: Critical Aspects

1. Manage trade-off between solution quality and computational effort (appropriate size of Kernel Set and buckets $\rightarrow$ size of subproblems)
2. Find alternatives to reduced costs when they do not provide a clear signal on promising variables

Kernel Search: Critical Aspects

1. Manage trade-off between solution quality and computational effort (appropriate size of Kernel Set and buckets $\rightarrow$ size of subproblems)
2. Find alternatives to reduced costs when they do not provide a clear signal on promising variables
3. Find an effective way for variables to enter/leave the Kernel Set after its initial construction

Kernel Search: Critical Aspects

1. Manage trade-off between solution quality and computational effort (appropriate size of Kernel Set and buckets $\rightarrow$ size of subproblems)
2. Find alternatives to reduced costs when they do not provide a clear signal on promising variables
3. Find an effective way for variables to enter/leave the Kernel Set after its initial construction
4. Optimize the breakdown of the global solution time used by the method

Kernel Search with ARC-PATH (KS1)

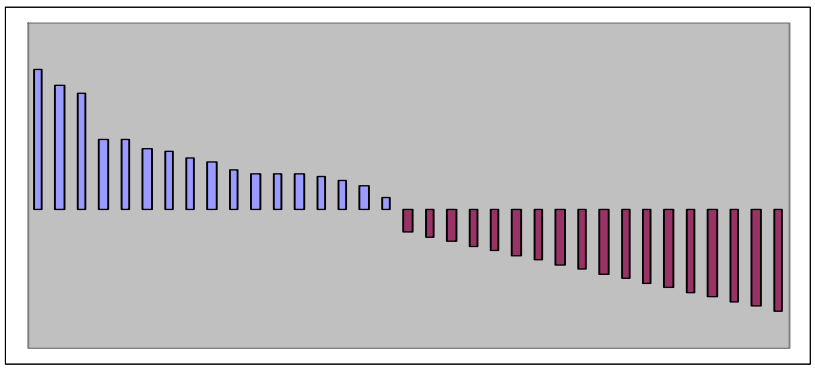
Iterative variant: the buckets are scrolled more than once (bucket iterations)

Innovative aspects:

1. Two variables sorting rules that can be interchanged in the bucket iterations: LP based SORTING and RPP Sorting
2. Dynamic procedure to eliminate variables from Kernel Set (using information from solved subproblems)
3. Redefinition of Kernel Set and bucket size at the beginning of each bucket iteration
4. Dynamic adjustment of subproblems solution time

LP RELAXATION based SORTING (LPS)

Sort the items (required arc, position, interval) according to variables and reduced costs value in the optimal solution of a Restricted LP relaxation:



Restricted LP relaxation:
Given the current two best solutions, solve an LP relaxation on the original graph, considering only the items belonging such solutions plus all the items that can be obtained by considering adjacent intervals and positions.
At each KS iteration, if the solution has improved, a new Restricted LP relaxation is solved.

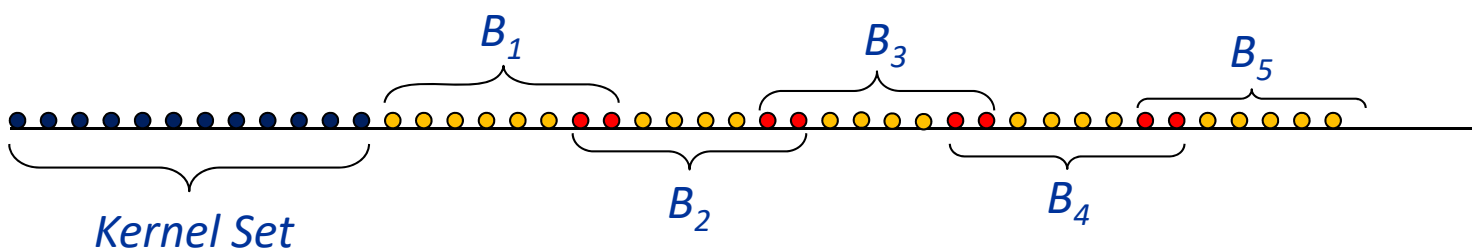
RPP based SORTING (RPPS)

- Generate several instances of RPP by randomly choosing, for each arc, one of its possible travel time;
- Solve the instances (by means of the Branch & Cut algorithm presented in Ávila, Corberán, Plana, and Sanchis (2015)) and sort them according to the value of their solution (after it has been transformed into a time-dependent solution);
- Items associated with the solutions are added to a bucket (if they have not been added yet) according to the rank assigned to the solution they belong to;
- items that have not been used in any solution are added to the buckets in random order.

Buckets and Kernel Set Construction

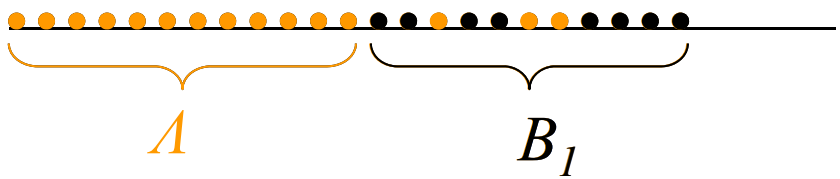
At the beginning of each bucket iteration:

- Kernel Set encompasses all items of the best integer solution found so far plus the first ϕ positions in the LPS
- Remaining items are divided into buckets of equal size (which increases at each new bucket iteration) that partially overlap:



Varying Kernel Set

- Insert procedure:
  - $\textsc{SolveSubproblem}(\Lambda \cup B)$ for each $B \in \mathcal{B}$
  - Add to Λ all clusters selected in the subproblem solution



- Eject procedure:
  - If $e_i - s_i \geq threshold$, remove item i from Λ

$\mathcal{S}_i = \{$set of solutions found by KS since item i entry in $\Lambda\}$

s_i (e_i) number of times item i is selected (not) by solutions in $\mathcal{S}_i$

Dynamic Adjustment of Time

For each iteration:

- Kernel Set: $(1+\beta)$ (Global time iteration)/(min#buckets+1)
- Each Bucket: $(1+\beta)$ (Residual time)/(residual min#buckets)

Motivation: Kernel Set and initial buckets should contain more promising items and thus corresponding subproblems should receive higher computational time

Initial Solution

All implemented variants of KS start from the same initial solution constructed as follows:

1. Set a time limit T_{max}
2. Solve the original instance by means of the Branch&Bound in the root node within a time T_{max}
3. Worst case scenario: the solution identified in the root node is used.

Exploiting Branch & Bound

We decided to try to use the Branch & Bound (Calogiuri et al. (2019)) to solve subproblems inside our Kernel Search

- PROS:
  - It is extremely efficient for small size problems
  - It can be used as a pure black-box
  - It can be modified at no cost to limit the allowed positions for each required arc in a path
- CONS:
  - It is neither straightforward nor efficient to limit the set of intervals for each arc.
  - ....

Kernel Search with Branch & Bound (KS2)

Same structure of KS1 with the following differences:

- KS is constructed using as ITEM the combination

$$(\underbrace{\text{REQUIRED ARC}}_{a \in R}, \underbrace{\text{POSITION IN THE PATH}}_{k})$$

- New sorting rules: POSITIONAL SORTING (PS) and APPROXIMATED TRAVEL TIME SORTING (ATTS)
- Variables sorting: Combination of normalized PS and ATTS.
- Subproblem construction: all possible intervals but a limited set of positions for each required arc
- MIP solver: Branch&Bound with a t_{max} time limit

POSITIONAL SORTING (PS)

Given a current solution with 4 required arcs:

$[(0,2) - \text{path} - (5,4) - \text{path} - (7,3) - \text{path} - (9,0)]$

Arc $(5,4)$ has:

- a positional distance of 1 from arcs $(0,2)$ and $(7,3)$
- a positional distance of 2 from arc $(9,0)$

Positional Sorting:

✓ Assign a penalty to each association (ARC, POSITION)

✓ the higher the positional distance of an arc from the position it holds in the current solution, the higher the penalty

APPROX TRAVEL TIME SORTING (ATTS)

Given a required arc in the current solution:

1. Given a position k of the required arc;
2. Sum up twice the traveling time of the $k-1$ required arcs that can be traversed at highest speeds;
3. identify the corresponding interval for the arc and its travel speed;
4. penalize the pair (ARC,POSITION) according to how much slower it is when compared to the best arc speed

Benchmark Problems

- Largest size instances in the literature $n = 60$ nodes and $m = 120$ arcs (Calogiuri et al. [2019])
- New 45 instances with larger size graphs generated as in Calogiuri et al. [2019]:
  - For each pair (n, m) and each percentage of required arcs we generate 5 instances:
    - 3 combinations of (nodes, arcs): $(60, 200), (90, 600), (120, 900)$
    - 3 values of required arcs percentage: 50%, 60% and 70%

Experimental Setting

- HW: 64-bit Windows Personal Computer, 2.93 GHz Intel i7 870 processor, and 8 GB of RAM.
- SW: Java 8, Gurobi 8.1.0: all built-in cuts activated, 8 threads, Barrier Algorithm for solving LP relaxation

Exact Algorithm: 1 hour time limit

– B&B: Branch-and-Bound by Calogiuri et al. [2019]

KS variants: 15 minutes time limit

– Iterative KS with combined positional/approximated travel time sorting

– Iterative KS with reduced costs sorting of Lp relaxation of arc-path formulation

Preliminary results: KS1 1/2

Name	Nodes	Arcs	% req	B&B	Gap%	KS1
Inst1	60	200	0.5	6580.57	9.02	6580.57
Inst2	60	200	0.5	8207.85	0.00	8207.85
Inst3	60	200	0.5	7054.36	16.36	7054.36
Inst4	60	200	0.5	7533.34	8.89	7533.34
Inst5	60	200	0.5	7529.18	9.37	7529.18
Inst6	60	200	0.6	9708.23	8.05	9708.23
Inst7	60	200	0.6	8954.31	8.23	**8951.12**
Inst8	60	200	0.6	8262.35	9.09	**8253.61**
Inst9	60	200	0.6	9974.92	10.62	**9961.36**
Inst10	60	200	0.6	10553.05	6.94	10553.05
Inst11	60	200	0.7	9772.90	8.27	**9762.79**
Inst12	60	200	0.7	9897.83	8.26	9897.83
Inst13	60	200	0.7	7591.70	9.65	**7558.76**
Inst14	60	200	0.7	8437.88	8.02	**8429.53**
Inst15	60	200	0.7	8155.26	8.29	**8146.76**
Inst16	90	600	0.5	21885.51	13.14	**21879.97**
Inst17	90	600	0.5	18755.86	15.13	18755.86
Inst18	90	600	0.5	21862.65	20.72	21862.65
Inst19	90	600	0.5	22239.27	20.76	22239.27
Inst20	90	600	0.5	19864.25	18.16	19864.25
Inst21	90	600	0.6	23114.53	17.58	**23107.65**
Inst22	90	600	0.6	20976.32	9.06	20976.32
Inst23	90	600	0.6	25299.77	12.44	**25286.31**

Preliminary results: KS1 2/2

Name	Nodes	Arcs	% req	B&B	Gap%	KS1
Inst24	90	600	0.6	21973.63	9.38	21973.63
Inst25	90	600	0.6	24561.59	8.86	24561.59
Inst26	90	600	0.7	22961.52	8.26	22961.52
Inst27	90	600	0.7	23176.13	9.04	23176.13
Inst28	90	600	0.7	25473.11	7.90	25473.11
Inst29	90	600	0.7	25020.85	10.22	25020.85
Inst30	90	600	0.7	23463.14	8.71	**23442.27**
Inst31	120	900	0.5	29849.50	19.61	29849.50
Inst32	120	900	0.5	28960.18	0.00	28960.18
Inst33	120	900	0.5	33279.22	10.88	33279.22
Inst34	120	900	0.5	31078.62	22.18	31078.62
Inst35	120	900	0.5	33310.35	20.10	33310.35
Inst36	120	900	0.6	35982.77	9.46	35982.77
Inst37	120	900	0.6	30474.84	9.12	30474.84
Inst38	120	900	0.6	29277.84	10.09	29277.84
Inst39	120	900	0.6	36660.45	8.68	36660.45
Inst40	120	900	0.6	29863.62	9.13	29863.62
Inst41	120	900	0.7	37043.95	8.83	37043.95
Inst42	120	900	0.7	36496.36	7.85	36496.36
Inst43	120	900	0.7	32976.25	9.04	32976.25
Inst44	120	900	0.7	37992.64	7.78	37992.64
Inst45	120	900	0.7	35950.46	9.65	35950.46

Preliminary results: KS2 1/2

Name	Nodes	Arcs	% req	B&B	Gap%	KS2
Inst1	60	200	0.5	6580.57	0.00	**6579.27**
Inst2	60	200	0.5	8207.85	6.94	8207.85
Inst3	60	200	0.5	7054.36	16.36	7054.36
Inst4	60	200	0.5	7533.34	8.89	7533.34
Inst5	60	200	0.5	7529.18	9.37	7529.18
Inst6	60	200	0.6	9708.23	8.05	**9694.02**
Inst7	60	200	0.6	8954.31	8.23	**8934.10**
Inst8	60	200	0.6	8262.35	9.09	**8243.11**
Inst9	60	200	0.6	9974.92	10.62	**9958.29**
Inst10	60	200	0.6	10553.05	6.94	**10542.00**
Inst11	60	200	0.7	9772.90	8.27	**9735.57**
Inst12	60	200	0.7	9897.83	8.26	**9871.49**
Inst13	60	200	0.7	7591.70	9.65	**7570.99**
Inst14	60	200	0.7	8437.88	8.02	**8434.28**
Inst15	60	200	0.7	8155.26	8.29	**8134.51**
Inst16	90	600	0.5	21885.51	13.14	**21884.25**
Inst17	90	600	0.5	18755.86	15.13	18755.86
Inst18	90	600	0.5	21862.65	20.72	21862.65
Inst19	90	600	0.5	22239.27	20.76	22239.27
Inst20	90	600	0.5	19864.25	18.16	**19859.05**
Inst21	90	600	0.6	23114.53	17.58	**23112.23**
Inst22	90	600	0.6	20976.32	9.06	**20972.60**
Inst23	90	600	0.6	25299.77	12.44	**25292.09**

Preliminary results: KS2 2/2

Name	Nodes	Arcs	% req	B&B	Gap%	KS2
Inst24	90	600	0.6	21973.63	9.38	21973.63
Inst25	90	600	0.6	24561.59	8.86	**24542.01**
Inst26	90	600	0.7	22961.52	8.26	**22945.21**
Inst27	90	600	0.7	23176.13	9.04	**23170.11**
Inst28	90	600	0.7	25473.11	7.90	**25438.05**
Inst29	90	600	0.7	25020.85	10.22	25020.85
Inst30	90	600	0.7	23463.14	8.71	**23446.47**
Inst31	120	900	0.5	29849.50	19.61	29849.50
Inst32	120	900	0.5	28960.18	0.00	28960.18
Inst33	120	900	0.5	33279.22	10.88	33279.22
Inst34	120	900	0.5	31078.62	22.18	31078.62
Inst35	120	900	0.5	33310.35	20.10	33310.35
Inst36	120	900	0.6	35982.77	9.46	**35949.76**
Inst37	120	900	0.6	30474.84	9.12	**30459.04**
Inst38	120	900	0.6	29277.84	10.09	29277.84
Inst39	120	900	0.6	36660.45	8.68	36660.45
Inst40	120	900	0.6	29863.62	9.13	**29858.37**
Inst41	120	900	0.7	37043.95	8.83	37043.95
Inst42	120	900	0.7	36496.36	7.85	36496.36
Inst43	120	900	0.7	32976.25	9.04	32976.25
Inst44	120	900	0.7	37992.64	7.78	37992.64
Inst45	120	900	0.7	35950.46	9.65	**35935.08**

Preliminary results: KS1 vs KS2

Name	B&B	KS1	KS2	Name	B&B	KS1	KS2
Inst1	6580.57	6580.57	**6579.27**	Inst24	21973.63	21973.63	21973.63
Inst2	8207.85	8207.85	8207.85	Inst25	24561.59	24561.59	**24542.01**
Inst3	7054.36	7054.36	7054.36	Inst26	22961.52	22961.52	**22945.21**
Inst4	7533.34	7533.34	7533.34	Inst27	23176.13	23176.13	**23170.11**
Inst5	7529.18	7529.18	7529.18	Inst28	25473.11	25473.11	**25438.05**
Inst6	9708.23	9708.23	**9694.02**	Inst29	25020.85	25020.85	25020.85
Inst7	8954.31	8951.12	**8934.10**	Inst30	23463.14	**23442.27**	23446.47
Inst8	8262.35	8253.61	**8243.11**	Inst31	29849.50	29849.50	29849.50
Inst9	9974.92	9961.36	**9958.29**	Inst32	28960.18	28960.18	28960.18
Inst10	10553.05	10553.05	**10542.00**	Inst33	33279.22	33279.22	33279.22
Inst11	9772.90	9762.79	**9735.57**	Inst34	31078.62	31078.62	31078.62
Inst12	9897.83	9897.83	**9871.49**	Inst35	33310.35	33310.35	33310.35
Inst13	7591.70	**7558.76**	7570.99	Inst36	35982.77	35982.77	**35949.76**
Inst14	8437.88	**8429.53**	8434.28	Inst37	30474.84	30474.84	**30459.04**
Inst15	8155.26	8146.76	**8134.51**	Inst38	29277.84	29277.84	29277.84
Inst16	21885.51	**21879.97**	21884.25	Inst39	36660.45	36660.45	36660.45
Inst17	18755.86	18755.86	18755.86	Inst40	29863.62	29863.62	**29858.37**
Inst18	21862.65	21862.65	21862.65	Inst41	37043.95	37043.95	37043.95
Inst19	22239.27	22239.27	22239.27	Inst42	36496.36	36496.36	**36485.80**
Inst20	19864.25	19864.25	**19859.05**	Inst43	32976.25	32976.25	32976.25
Inst21	23114.53	**23107.65**	23112.23	Inst44	37992.64	37992.64	37992.64
Inst22	20976.32	20976.32	**20972.60**	Inst45	35950.46	35950.46	**35935.08**
Inst23	25299.77	**25286.31**	25292.09				

Conclusions and Future Developments

Already obtained:

- an effective variant of KS for solving larger size TDRPP
- a partial modification of the Branch-and-Bound algorithm

Conclusions:

- KS1: sorting by using the LP relaxation is a valuable option
- KS2: Branch & Bound is the best choice to solve subproblems

To be done:

- strong refinement of the Branch & Bound to solve constrained subproblems with restricted number of positions and time intervals for each arc
- a combined version of the two variants