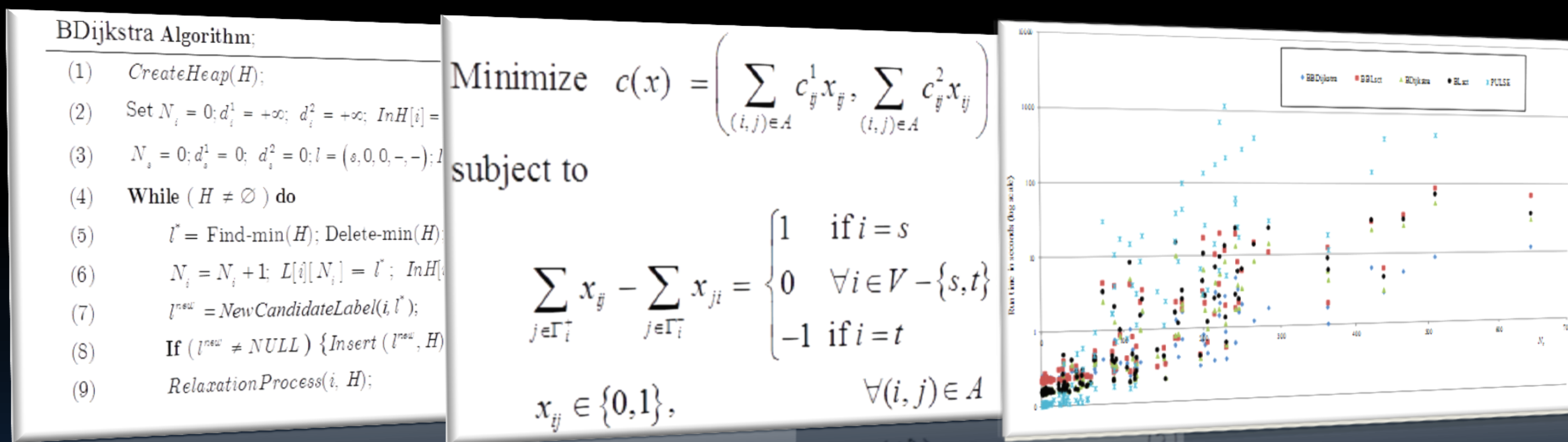
The Multiobjective Dijkstra's Algorithm



Antonio Sedeño-Noda & Marcos Colebrook
Department of Mathematics, Statistics and Operations Research
Department of Informatics and Systems
Universidad de La Laguna

Introduction

- In this work, we investigate the existence of a generalized Dijkstra's algorithm to compute non-dominated solutions of the one-to-all problem.
- The features of the proposed algorithm are:
  - **Keeps only one candidate label per node in a priority queue of size at most the number of nodes n.**
  - Needs a function to obtain the next candidate label per node.
  - Simplify all the basic operations in the label-setting method. For example it does not need ***merge*** operations among set of labels.
- We consider a unidirectional and bidirectional version of this algorithm.
- We observe the results of a computational experiment including others state-of-the-art BSP methods.

Problem formulation

Directed Network $G = (V, A)$ with n nodes and m arcs

- A origin (source) node s. A destination (target) node t.
- Each arc (i, j) has two real costs $c_{ij} = (c^1_{ij}, c^2_{ij})$.

- The length of a directed path P from node s to node i is the sum of the length arcs in the path P;
- **The one-to-all *Biobjective Shortest Path* problem**

$$\text{Minimize} \quad c(x) = \left(\sum_{(i,j)\in A} c^1_{ij} x_{ij}, \sum_{(i,j)\in A} c^2_{ij} x_{ij} \right) \qquad (1)$$

subject to

$$\sum_{j\in\Gamma_i^+} x_{ij} - \sum_{j\in\Gamma_i^-} x_{ji} = \begin{cases} n-1 \text{ if } i = s \\ -1 \quad \forall i \in V - \{s\} \end{cases} \qquad (2)$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A \qquad (3)$$

Assumptions & definitions

- **Assumption 1.** The network G contains a directed path from origin node s to any node i in V. (w.l.o.g.)
- **Assumption 2.** The network G does not contain a directed cycle with negative length for each single-objective SP problem.
  - Then, we consider (w.l.o.g) that real costs $c_{ij} = (c^1_{ij}, c^2_{ij})$ takes non-negative values.
  - If ***G*** contains arcs with negative costs, we can work with reduced costs:

$$\bar{c}^v_{ij} = c^v_{ij} + \pi^v_i - \pi^v_j \geq 0 \ \forall (i,j) \in A, v = 1,2$$

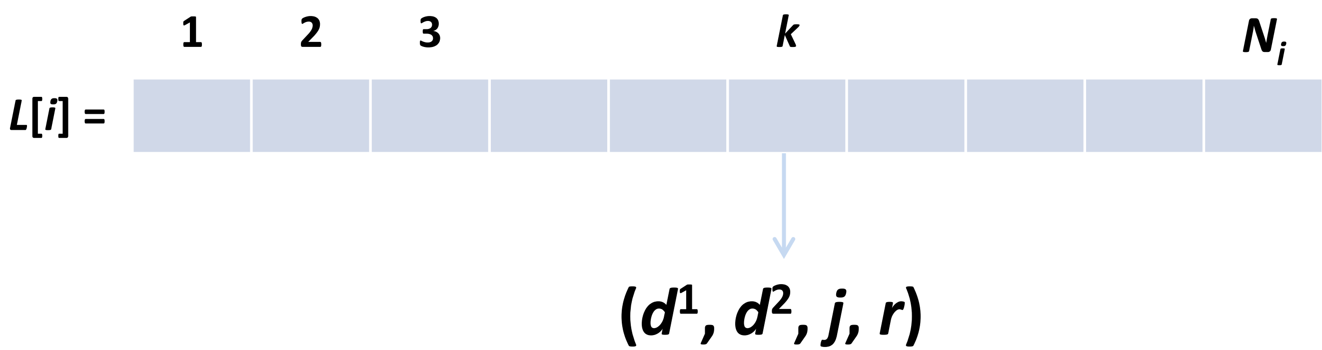
Assumptions & definitions

- **Definition 1.** A path (feasible solution) p_1 in P (x_1 in X) is called ***efficient*** if there does not exist any p_2 in P (x_2 in X) with $c^1(p_1) \leq c^1(p_2)$ and $c^2(p_1) \leq c^2(p_2)$ with at least one inequality being strict. The image $c(p)$ ($c(x)$) of an efficient path p (or solution x) is called ***non-dominated point.***
- **Proposition 1** (Martins, 1984): **Principle of Optimality.** Every efficient path p from node s to node t contains only efficient sub-paths from s to any intermediate node in p .
- **Definition 2.** *Lexicographic order* ($\prec_L$) in $\Re^2$.

  Let (a, b) and (c, d) be two points in $\Re^2$. We say that (a, b) is lexicographically smaller than (c, d), denoted by $(a, b) \prec_L (c, d)$, if $a < c$ or ($a = c$ and $b < d$) holds.

Labels & methods

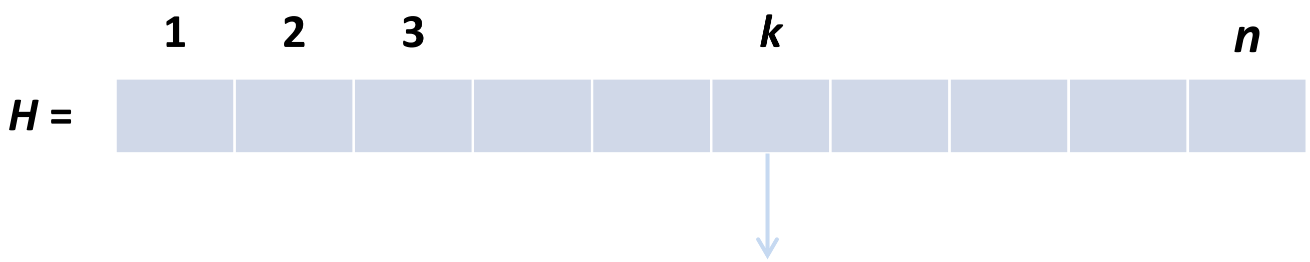
- The main purpose of a label-setting method is compute for all node i in V, **the set of non-dominated (permanent) labels $L[i]$**



- N_i is the number of non-dominated points of the one-to-one BSP problem from node s to node i.
- (d^1, d^2, j, r) is the kth label in $L[i]$, where:
  - d^1, d^2 are the distances in a path P from node s to node i.
  - j is the **predecessor node** of node i in P.
  - r is the ***position*** in $L[j]$ ($L[j][r]$) containing the label from the kth label in $L[i]$ was calculated.

Labels & methods

- Classical label-setting methods use a set ***T*** of temporary labels (not yet expanded).
- Instead, we use a **priority queue *H* of maximum *size n.***



$$l = (i, d^1, d^2, j, r)$$

Invariant 1. Our algorithm keeps the invariant that the label ***l*** in ***H*** associated with node *i* is not dominated by any label in ***L*[*i*]** and ***l*** is not in ***L*[*i*]**, for all *i* in *V.*

Additionally, we use a bool vector ***inH*** where ***inH*[*i*]** is true if heap ***H*** contains a label associated with node *i*, and false otherwise.

The BDijkstra Algorithm: Selection criterion

$$l^* = \underset{l \in H}{\arg \operatorname{lex} \min} \left\{ (l.d^1, l.d^2) \right\}$$

The algorithm determines all non-dominated labels of the one-to-all BSP problem in lexicographic min order.

$N_i = N_i + 1$

1 2 3 k N_i

$L[i]$ = [| | | | | | | | | | l^*]

$l^* =$ Find-min(H); Delete-min(H); This operation takes $O(\log n)$ time

Add l^* at the end of $L[i]$; $N_i = N_i + 1$;
$InH[i] = False$; This operation takes $O(1)$ time

// i is the node with label l^*

l^* is a definitive (permanent) label in $L[i]$ **(Theorem 1)**.

The BDijkstra Algorithm: Determination of a new label

The next label for node *i* must be the lexicographic smallest non-dominated label among the labels that can be obtained from the labels in *L*[*j*], for any node *j* predecessor of node *i*.

$$l^{new} = \arg \underset{\forall j \in \Gamma_i^-, \forall l \in L[j]}{\text{lex}\min} \left\{ (l.d^1 + c_{ji}^1, l.d^2 + c_{ji}^2) \mid l.d^1 + c_{ji}^1 > l^*.d^1 \text{ and } l.d^2 + c_{ji}^2 < l^*.d^2 \right\}$$

l^{new} non-dominated by l^* & $l^* \prec_L l^{new}$

Theorem 2. *Let G = (V, A) be directed network with non-negative cost vector* $c_{ij} = (c_{ij}^1, c_{ij}^2)$ *for all arcs* $(i, j) \in A$. *In the BDijkstra algorithm, the dominance test on a label distance* (a,b) *for a node i needs only the last label in L[i].*

The BDijkstra Algorithm: Determination of a new label

Function *NewCandidateLabel*$(i,\ l^*)$

$d^1 = +\infty; d^2 = +\infty; l^{new} = Null;$

For $j \in \Gamma_i^-$ **do**

  For $l \in L[j]$ **do**

    If $((l.d^1 + c_{ji}^1 < d^1)$ **or** $(l.d^1 + c_{ji}^1 == d^1$ **and** $l.d^2 + c_{ji}^2 < d^2))$ //lexmin candidate label

      If $(l.d^1 + c_{ji}^1 > l^*.d^1$ **and** $l.d^2 + c_{ji}^2 < l^*.d^2)$ //non-dominated candidate label

        $d^1 = l.d^1 + c_{ji}^1;\ d^2 = l.d^2 + c_{ji}^2;$

        $l^{new} = \left(i, d^1, d^2, j, r\right);$

        // r is the position of l in $L[j]$

Returns l^{new};

The function ***NewcandidateLabel*** takes a time of $O\left(\sum_{j\in\Gamma_i^-} |L[j]|\right)$

The BDijkstra Algorithm: updating the labels in *H*. Relaxation.

Procedure *RelaxationProcess*(*i*, *H*)

For all $j \in \Gamma_i^+$ **do**

  If $((l^*.d^1 + c_{ij}^1 < d_j^1)$ **or** $(l^*.d^1 + c_{ij}^1 == d_j^1$ **and** $l^*.d^2 + c_{ij}^2 < d_j^2))$ //relaxation (i, j)

    If $(l^*.d^1 + c_{ij}^1 > L[j][N_j].d^1$ **and** $l^*.d^2 + c_{ij}^2 < L[j][N_j].d^2)$ **or** $(N_j == 0)$ //non-dominated label

      $d_j^1 = l^*.d^1 + c_{ij}^1$; $d_j^2 = l^*.d^2 + c_{ij}^2$;

      $l = (j, d_j^1, d_j^2, i, r)$;// r is the position of l^* *in* $L[i]$

      If $(InH[j]{==}False)$

        $\{Insert(l,\ H);\ InH[j]{=}True;\}$

      Else *decrease-key*$(l,\ H)$;

This part practically equals the relax operation in single-objective Dijkstra's algorithm.

The computational effort is O($|\Gamma_i^+|$) when *H is a Fibonacci heap*.

The BDijkstra Algorithm

BDijkstra **Algorithm**;

(1) $CreateHeap(H)$;

(2) Set $N_i = 0; d_i^1 = +\infty;\ d_i^2 = +\infty;\ InH[i] = False;$ for all $i \in V - \{s\}$;

(3) $N_s = 0; d_s^1 = 0;\ d_s^2 = 0; l = (s, 0, 0, -, -); Insert(l, H); InH[s] = True;$

(4) **While** ($H \neq \varnothing$) **do**

(5) $l^* =$ Find-min(H); Delete-min(H); — $O(N\log n)$ time in overall with $N = \sum_{i=1}^{n} N_i$

(6) $N_i = N_i + 1;\ L[i][N_i] = l^*;\ InH[i] = False;$ // i is the node with label l^*

(7) $l^{new} = NewCandidateLabel(i, l^*)$; — $O(mN_{max})$ time in overall with $N_{max} = \max_{i \in V}\{N_i\}$

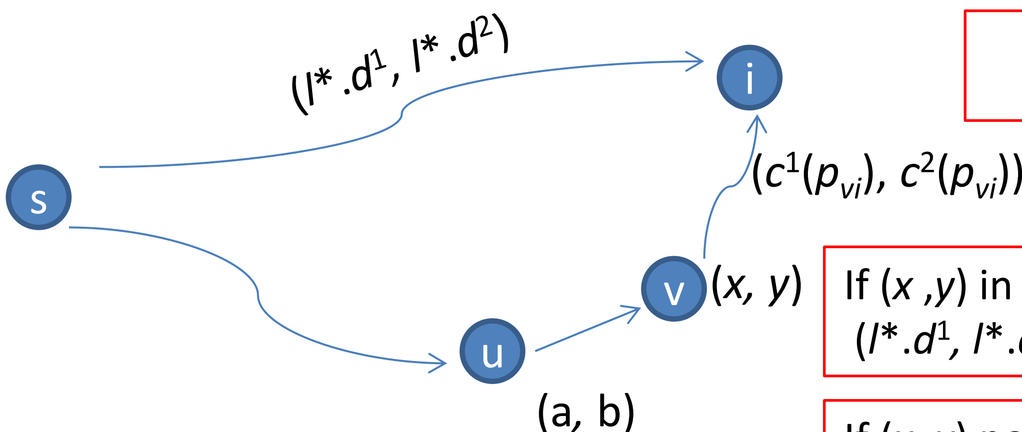
(8) **If** ($l^{new} \neq NULL$) $\{Insert\ (l^{new}, H);\ InH[i] = True; d_i^1 = l^{new}.d^1; d_i^2 = l^{new}.d^2;\}$

(9) $RelaxationProcess(i, H)$; — $O(mN_{max})$ time in overall

Theorem 3. *The BDijkstra algorithm runs in* $O(N \log n + mN_{max})$ *time and uses* $O(N+m+n)$ *space.*

Correctness of the BDijkstra Algorithm

Theorem 1. *Let G = (V, A) be directed network with non-negative cost vector* $c_{ij} = (c_{ij}^1, c_{ij}^2)$ *for all arcs* $(i, j) \in A$. *The label* $l^* = \arg \operatorname{lex} \min_{l \in H} \{(l.d^1, l.d^2)\}$ *corresponds to an efficient path from node s to node* i *, that is, there is no path p from s to i in G whose distance* $(c^1(p), c^2(p))$ *dominates* $(l^*.d^1, l^*.d^2)$.



The alternative path has length
$(c^1(p), c^2(p)) = (x, y) + (c^1(p_{vi}), c^2(p_{vi})) \geq (x, y)$

If (x, y) in ***H*** then
$(l^*.d^1, l^*.d^2) \prec_L (x, y) \prec_L (c^1(p), c^2(p))$

If (x, y) not in ***H*** then, must exist (x', y') in ***H*** with $(x', y') \prec_L (x, y)$ and, therefore

$(l^*.d^1, l^*.d^2) \prec_L (x', y') \prec_L (x, y) \prec_L (c^1(p), c^2(p))$

u *is the node in the path* ***p*** *such that its label* (a, b) *is in* **L**[***u***], but the *label* (*x*, *y*) *of its successor node in the path is not in* **L**[**v**].

Solving the one to one BSP problem

1. Computing only the necessary non-dominated labels. Pruning strategies.

- Skriver, A. J. V., Andersen K. A. (2000). A label correcting approach for solving bicriterion shortest-path problems. *Computers and Operations Research*, 27, 507-524.
- Duque, D., Lozano, L., Medaglia, A. L. (2015). An exact method for the biobjective shortest path problem for large-scale road networks. European Journal of Operational Research, 242 (3), 788-797.

2. Bidirectional scheme. The BBDijkstra algorithm.

- Single- objective case.

- Pohl, I. (1971). Bi-directional Search. In Machine Intelligence 6, 124–140. Edinburgh Univ. Press, Edinburgh.

- Multiobjective case.

- Demeyer, S., Goedgebeur, J., Audenaert, P., Pickavet, M., Demeester, P. (2013). Speeding up Martins' algorithm for multiple objective shortest path problems. 4OR - A Quarterly Journal of Operations Research, 11 (4), 323–348.

Computational Results: Algorithms

- Unidirectional Algorithms:
  - The PULSE algorithm given by Duque *et al*. (2015). **PULSE**.
  - Classical label-setting biobjective algorithm. **BLset**.
  - **BDijkstra**.

- Bidirectional Algorithms:
  - Classical label-setting biobjective algorithm. **BBLset**.
  - **BBDijkstra**.

- All algorithms include the pruning strategies.
- BLSet & BBLSet including the stop criterion in Demeyer *et al*. (2013) within or without the pruning strategies are slower than BLset & BBLset only including the pruning strategies.

Computational Results: Instance sets

Name	*n*	*m*
NY	264346	733846
BAY	321270	800172
COL	435666	1057066
FLA	1070376	2712798
NE	1524453	3897636
CAL	1890815	4657742
LKS	2758119	6885658

Road Networks from the DIMACS Shortest Path Implementation Challenge (2013).
We have considered 100 different origin-destination pairs in each Road.

Name	Height	Width	*n*	*m*
G1-G7	300	300,350,…,600	90002 to 180002	35940 to 718800
G8-G14	350	300,350,…,600	105002 to 210002	419400 to 838800
G15-G21	400	300,350,…,600	120002 to 240002	479400 to 958800
G22-G28	450	300,350,…,600	135002 to 270002	539400 to 1078800
G29-G35	500	300,350,…,600	150002 to 300002	599400 to 1198800
G36-G42	550	300,350,…,600	165002 to 330002	659400 to 1318800
G43-G49	600	300,350,…,600	180002 to 360002	719400 to 1438800

Grid Networks. We have considered $s=1$ and $t=n$ in each grid.

Computational Results: Road Networks

		N_t	BBDijkstra	BBLset	BDijkstra	BLset	PULSE
NY	Solved/100	100/100	100/100	100/100	100/100	100/100	98/100
	Avg.	119.79	**0.74**	4.17	1.32	3.78	119.71
	Min	1	0.22	0.20	0.15	0.14	0.10
	Max	646	8.46	84.84	21.75	69.48	1137.92
BAY	Solved/100	100/100	100/100	100/100	100/100	100/100	98/100
	Avg.	143.77	**1.28**	5.08	2.16	6.29	234.87
	Min	1	0.27	0.26	0.18	0.19	0.11
	Max	825	16.39	99.92	33.42	102.38	2714.16
COL	Solved/100	100/100	100/100	100/100	100/100	100/100	86/100
	Avg.	346.51	**10.89**	47.01	12.20	39.08	657.88
	Min	1	0.36	0.35	0.25	0.24	0.16
	Max	2612	255.25	1087.77	355.57	1126.05	2442.59
FLA	Solved/100	100/100	100/100	98/100	99/100	97/100	69/100
	Avg.	673.72	**83.86**	314.32	129.21	310.28	1219.87
	Min	2	0.93	0.85	0.62	0.62	0.40
	Max	6292	1596.66	3559.18	1626.24	3477.06	2349.48
NE	Solved/100	99/100	99/100	93/100	99/100	97/100	49/100
	Avg.	808.19	246.95	581.36	**199.83**	516.28	1960.60
	Min	7	1.56	1.52	1.05	1.00	0.62
	Max	3145	3414.14	2873.28	1308.06	3496.64	2063.56
CAL	Solved/100	99/100	98/100	93/100	98/100	94/100	58/100
	Avg.	862.54	**216.99**	654.46	267.29	587.30	1706.22
	Min	1	1.90	1.79	1.24	1.32	0.73
	Max	6962	2543.59	3466.02	2786.61	3438.27	2247.70
LKS	Solved/100	74/100	68/100	55/100	74/100	56/100	25/100
	Avg.	1917.80	1577.87	1959.56	**1421.14**	1900.95	2805.81
	Min	1	2.93	2.90	1.92	2.00	1.09
	Max	7547	3560.20	3284.93	3286.70	3047.47	3186.06

[a] Average time is calculated with a computational time of 3600 seconds for unsolved instances.
[b] Maximum computational time of the solved instances.

Computational Results: Road Networks

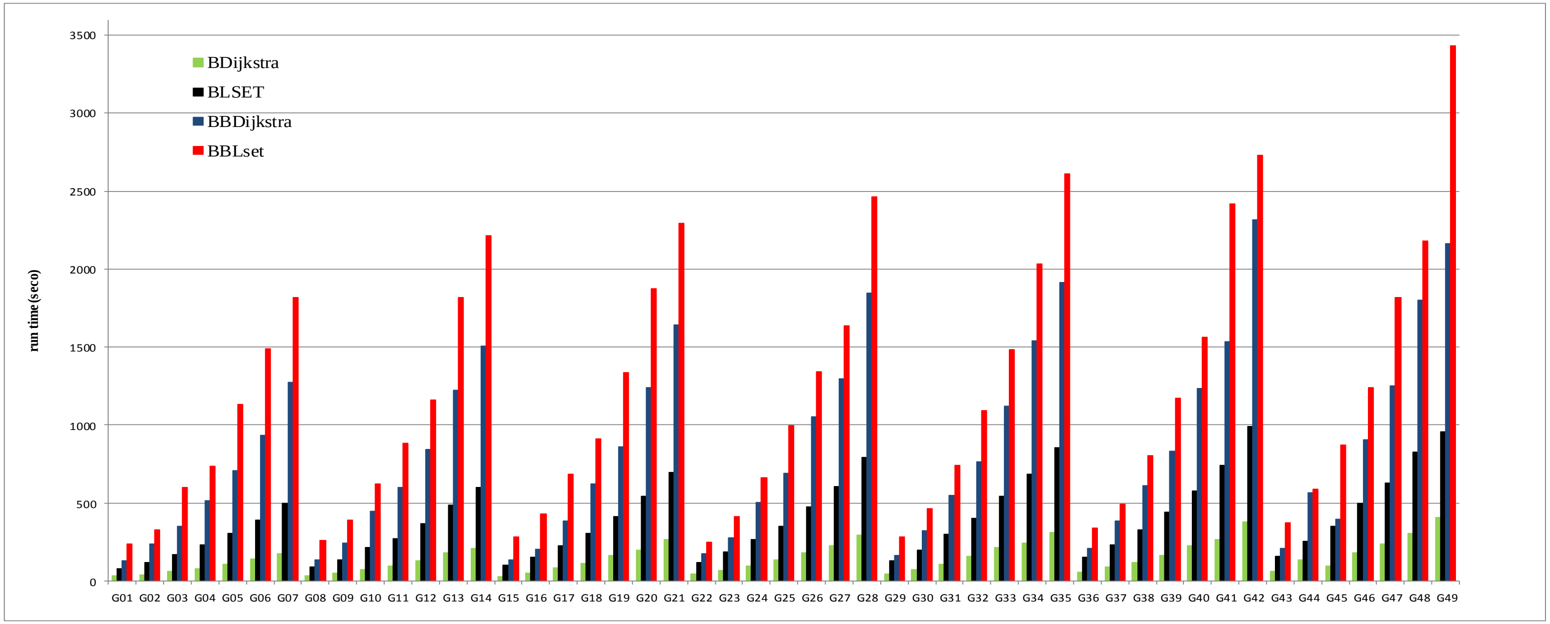
		N_t	BBDijkstra	BBLset	BDijkstra	BLset	PULSE
NY	Solved/100	100/100	100/100	100/100	100/100	100/100	98/100
	Avg.	119.79	**0.74**	4.17	1.32	3.78	119.71
	Min	1	0.22	0.20	0.15	0.14	0.10
	Max	646	8.46	84.84	21.75	69.48	1137.92
BAY	Solved/100	100/100	100/100	100/100	100/100	100/100	98/100
	Avg.	143.77	**1.28**	5.08	2.16	6.29	234.87
	Min	1	0.27	0.26	0.18	0.19	0.11
	Max	825	16.39	99.92	33.42	102.38	2714.16
COL	Solved/100	100/100	100/100	100/100	100/100	100/100	86/100
	Avg.	346.51	**10.89**	47.01	12.20	39.08	657.88
	Min	1	0.36	0.35	0.25	0.24	0.16
	Max	2612	255.25	1087.77	355.57	1126.05	2442.59
FLA	Solved/100	100/100	100/100	98/100	99/100	97/100	69/100
	Avg.	673.72	**83.86**	314.32	129.21	310.28	1219.87
	Min	2	0.93	0.85	0.62	0.62	0.40
	Max	6292	1596.66	3559.18	1626.24	3477.06	2349.48
NE	Solved/100	99/100	99/100	93/100	99/100	97/100	49/100
	Avg.	808.19	246.95	581.36	**199.83**	516.28	1960.60
	Min	7	1.56	1.52	1.05	1.00	0.62
	Max	3145	3414.14	2873.28	1308.06	3496.64	2063.56
CAL	Solved/100	99/100	98/100	93/100	98/100	94/100	58/100
	Avg.	862.54	**216.99**	654.46	267.29	587.30	1706.22
	Min	1	1.90	1.79	1.24	1.32	0.73
	Max	6962	2543.59	3466.02	2786.61	3438.27	2247.70
LKS	Solved/100	74/100	68/100	55/100	74/100	56/100	25/100
	Avg.	1917.80	1577.87	1959.56	**1421.14**	1900.95	2805.81
	Min	1	2.93	2.90	1.92	2.00	1.09
	Max	7547	3560.20	3284.93	3286.70	3047.47	3186.06

[a] Average time is calculated with a computational time of 3600 seconds for unsolved instances.

Computational Results: Grid networks

Algorithm	Average CPU Time	Max CPU time	Min CPU time	#Solved	Average N_t	Min N_t	Max N_t
BBDijkstra	833.74	2315.50	126.62	49	706.29	403	991
BBLset	1183.42	3431.30	237.38	49	706.29	403	991
BDijkstra	**147.12**	408.25	27.48	49	706.29	403	991
BLset	395.88	990.46	78.00	49	706.29	403	991
PULSE	3600.00			0			

[a] Average time is calculated with a computational time of 3600 seconds for unsolved instances.



Conclusions and future work

CONCLUSIONS:

- BBDijkstra and BDijkstra are the best in Road Networks.
- BDijkstra is the best in Grid instances.
- Consider one candidate label per node only is a good choice.
- Simplify the algorithm and allow us to determine a Theoretical Time Complexity. The space used is minimal.

FUTURE WORK:

- To extend to the Multiobjective case with an improved dominance-test in practice.
- Parallelize BDijkstra and BBDijkstra.

Questions?