Efficient Optimal Search under Expensive Edge Cost Computation

Masataro Asai, Akihiro Kishimoto, Adi Botea, Radu Marinescu , Elizabeth M. Daly and Spyros Kotoulas
The University of Tokyo, IBM Research Ireland

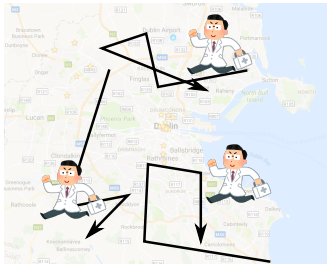
Motivation: Domains where edge-costs are not known apriori

Example 1: TSP w/ unknown edge costs

- O(N2) edge cost computations
- N: number of cities
- TSP itself is NP-complete

Example 2: Multiple Worker Routing Problem (MWRP)

A problem of finding the journey plans for workers (doctors, nurses) to attend the patients across the city, using multi-modal journey (using buses, trains, walking, etc...)



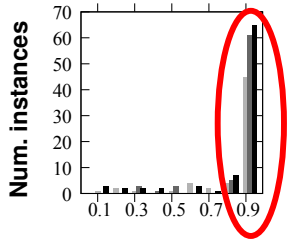
- $O(N^2T)$ edge cost computations
- N: number of locations
- T: number of possible start time (e.g. 1 day = 24x60x60 sec)
- Optimally solving MWRP is NP-hard (by reduction from SMTTP [Du and Leung, 1990])

city	nodes	segments	stops	routes	trips/day
Dublin	301,638	319,846	4,739	120	7,308
Montpellier	152,949	161,768	1,297	36	3,988
Rome	522,529	566,400	8,896	391	39,422

Table 1: Statistics of transport data used in experiments.

Trivial approach with A*: Compute the edge costs on demand

- Compute the edge costs during the search
- Evaluate the edge cost using an external solver DIJA [Botea et al., 2013] when computing the g() value of A*
- The result of solving DIJA is cached (the same edge is never computed twice)



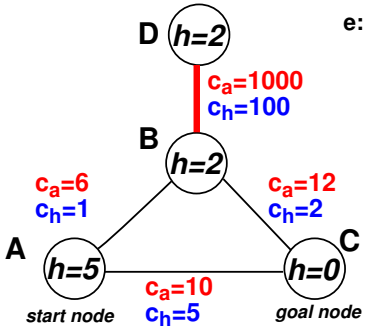
Fraction of runtime for the edge cost calcurations in the standard A* solving MWRP

- **Even with caching, most of the runtime is spent on calling DIJA solver**

Delayed Expansion A* (DEA*):

- The **actual edge cost c_a** is too expensive to compute. In many cases, c_a is unnecessary.
- Delay the evaluation of c_a by using the **lower-bound of the edge cost, c_h,** provided by the external solver
- Insert the same node twice, f-value computed once by c_h, then by c_a

Pathological case for A*: DEA* can avoid evaluating the edge BD



e: expand d: deleteMin i: insert

A*	DEA*	
i A(5)	i A(5)	e B(8)
d A(5)	d A(5)	i D(106)
e A(5)	i A(5)	i C(8)
i B(8)	d A(5)	d C(8)
i C(10)	e A(5)	i C(18)
d B(8)	i B(3)	d C(10)
e B(8)	i C(5)	e C(10)
i D(1006)	d B(3)	goal
i C(18)	i B(8)	
d C(10)	d C(5)	
e C(10)	i C(10)	
goal	d B(8)	

Evaluating DEA* on MWRP

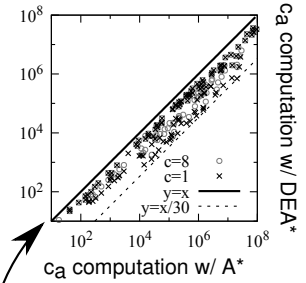
A*/DEA* using Blind / h^{max} / h^2 heuristics

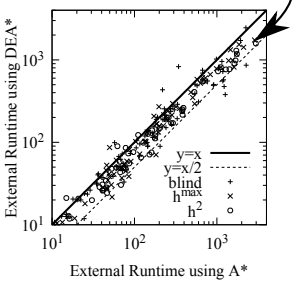
DEA* with h^1h^2: use h^{max} in the first expansion, use h^2 in the second expansion

City	worker |W|	Blind A*	Blind DEA*	h^{max} A*	h^{max} DEA*	h^2 A*	h^2 DEA*	h^1h^2 DEA*
Total		50	**59**	68	**77**	81	**83**	**85**
Dublin	12	14	**15**	15	**21**	19	**22**	**22**
	6	2	**4**	8	8	12	12	11
Montpellier	12	16	15	17	17	17	15	**18**
	6	1	**4**	3	**5**	3	**4**	**5**
Rome	12	15	**18**	18	**19**	19	19	**20**
	6	2	**3**	7	7	11	11	9

Number of solved instances

- Improved overall performance
- In some cases x2 OPEN list operation can be a bottleneck, losing to A*
- Time spent for calling DIJA has been reduced
- Results in the smaller overall runtime





DEA* on Classical Planning

- Evaluate the domain-independent performance
- Standard IPC benchmark assumes the edge costs are given, so we "simulate" the case where c_a is expensive & calculated dynamically
- Simulated DEA*: $c_h = c_a - C$, C: constant value
- DEA* reduced the c_a computation (up to x30)