A GLANCE ON HYPERPATH

HyperPath is a complete software solution for personal routing on multimodal transport networks with road infrastructures and transit line services. It is a highly powerful and flexible journey planner that allows users to find their optimal path between two given points in space, within both urban and intercity contexts. HyperPath is an on-line system, specifically designed for intermodal trips, to provide a web service for the personalized computation in real-time of the best route from an origin to a destination, in a given day and time, taking into account the estimated, current or forecasted traffic conditions and any source of service availability information.

The system can be accessed through the web, also from smart-phone dedicated pages. It is made by two main components: a web client application (developed in php and javascript), that allows the final user to formulate via internet his/her path requests through any (current) browser and to display as well as navigate the resulting route indications in text and map forms; a server core application (developed in VisulBasic.Net), that receives the path requests and processes them through sophisticated extremely performing algorithms.

On the server side, HyperPath is a dynamic routing engine for searching and describing the optimal journey strategy on an intermodal transport network, in presence of travel times and costs that vary during the day, due to congestion or service availability. Given a desired departure time, HyperPath finds a dynamic shortest route in terms of generalized cost, possibly with personalized weights for each path request, by means of an exact, but extremely efficient, shortest path algorithm, developed ad-hoc. Specifically, it takes into account in the optimization the estimated/forecasted performances that will occur at the actual time when the user will travel on each link of the network. HyperPath then describes the resulting itinerary through a suitable sequence of textual travelling instructions, each one associated with a geometrical entity on a reference cartography.

On the client side, HyperPath is a user-friendly web application to support the submission of a path request, that allows entering all the required parameters besides origin and destination, such as the departure day and time or any travelling preferences. The solution, that is in general a sequence of road links and transit lines, can then be easily navigated on the map exploiting the correspondence for each suggested action (e.g. turn left on that street, walk for 500 meters on that road, board that line at that stop) between its textual instruction and geometrical entity.

HyperPath includes also a software module for geocoding and reversed geocoding, that allows for disambiguation of address and POI, when providing the origin and destination of the trip or any waypoint. It has a free textual search, where the user can input all the information without a specific order. The input text is then automatically translated into geographic coordinates to identify the corresponding point on the street map. In presence of ambiguities, for homonyms or typos, the system proposes a set of possible alternatives, among which the user can choose the desired one. As an alternative, the user can specify directly each trip terminal by clicking a point on the digital map; in this case reversed geocoding is applied to identify the closest address to the selected point.

No fancy hardware is required; a dedicated standard machine running the travel planner application, including the search engine and the web site, is enough. In the following we describe in detail the main features and functionalities of the product.

FEATURES AND FUNCTIONALITIES OF HYPERPATH

Road network with navigation details. The road network can be fully described with turn prohibitions and access limitations, in addition to speed and intersection delay for each directional link. Mode specific tolls and costs can be easily introduced where needed.

Truck routing attributes and weights. The software is capable of handling a wide range of link attributes that are important for the routing of heavy vehicles, such as: tortuosity, steepness, bridges, tunnels, left and right turns at intersections, urban context, change of street hierarchy level, major roads. Each attribute can be scaled by a personalized weigh in the computation of generalized costs.

Transit networks, both frequency and scheduled based. The transit line service can be characterized, given the stop sequence, in terms of headway distribution and travel times (frequency-based), or in terms of single runs (schedule-based). For example, at each stop we can deal at the same time and in the same framework with a given line passing on average every five minutes, and with a specific run departing at 8:00.

Pedestrian network. The topology of the pedestrian network can be (as usual) introduced explicitly together with the road network. Different speeds can be defined for each link to simulate "fast pedestrians" with the aim of reproducing in a simple way the access to commuter transit stops by private vehicles or secondary bus lines. This way it is possible to avoid the explicit representation of the auxiliary network links and transport modes, so as to simplify the data model and speed-up the routing algorithm.

Import plug-ins and formats. The road network can be imported directly from Navteq data. The transit network can be specified in the Visum format or in the Maior format.

Flexible map projections. Both the background images and the network data can be projected in the same map framework, to improve the orientation of the user during the input-request phase and enrich the presentation of results during the output-route phase.

Transit travel times easily provided. In case of frequency based service, the travel times can be: described independently for each section between stops (for example based on historic GPS data of carrier trajectories); based on the road travel time plus a dwelling time at stops (this may be the case of shared road use by public and private transport), simply derived through a commercial speed. This allows to set-up the base model with little efforts exploiting directly the available information.

Transit data via service schedule. As an alternative, all the necessary line attributes (e.g. frequencies and travel times; both varying during the day) can be automatically retrieved from the programmed schedule of the service, i.e. the data source typically available at transit companies.

Regularity besides frequency. Each transit line can be characterized in terms of headway distribution, which means that the simulated waiting times are not necessarily (as usual) exponential, but may be also uniform. This allows to take into account in the routing algorithm the regularity of the service besides its total amount.

Fare structure. Kilometric or boarding fees and costs can be explicitly considered, while more complex fare structures (areas, belts, sections) can be introduced whenever reducible to a zone based (origin-destination) matrix.

Multiple modes. Based alternatively on the road network or on the transit plus pedestrian network, it is possible to define different modes with their own tolls and costs. A maximum speed can be specified to define different private modes, such as bike and motorcycles, as an alternative to introduce dedicated links.

Multiple classes. Each mode embeds also specific weights for different cost components, such as the value of time, the multiplier of walking and waiting times, additional costs for transfers on the transit network and change of hierarchy on the road network. This allows to define different classes in the route choice, each one with its own bundle of travelling preferences. Users can select a class when submitting their path request, e.g. fast, simple, little walking, low cost, etc.

Personalized travelling preferences. To fully meet the needs of the more demanding users, the software provides also the possibility of specifying manually the values of the cost weights, for each path request. This requires in the algorithm to compute the link costs “on the fly”, but permits the complete personalization of search and a sensitivity analysis approach to look for alternative routes with a similar cost to the best solution.

Park and Ride. Optimal routing on intermodal trips which require to park a private vehicle at a commuting terminal is ensured by a specialized search procedure. Time-varying delays and costs of each parking facility can be explicitly taken into account.

Passenger strategies. The routing method used for frequency based networks is natively conceived to deal with the concept of travel strategy, where a passenger at a transit stop can board the first arriving carrier of an attractive line set to reach his destination. This modelling approach is very appropriate in the case of common (partially overlapping) lines, where the passenger can board any of them to reduce his waiting time. Here it is extended to alternative paths, thus leading to the idea of hyperpath.

Implicit connection among stops. If in the available dataset the nodes of the transit stops coincide with the nodes of the road links, then the connection among stops is realized explicitly through the pedestrian network. However, this is often not the case when the public and private graphs come from different sources. We thus developed the idea of implicit connections: if closer than a given threshold (e.g. 300 meters) two stops are connected automatically within the routing algorithm without modifying the dataset.

Actual trip terminals and geocoding. The origin and destination of the trip are, as usual, provided by the traveller via a free text in Google style. Street addresses or point of interests are both allowed. The Google api are indeed used to geocode this information and provide the trip terminals in terms of points on the map. As an alternative, the user can specify directly each trip terminal by clicking a point on the map; in this case reversed geocoding is applied to identify the closest address to the selected point. These geographic coordinates are then sent to the server application. If a street database is provided, HyperPath can even use its own proprietary module for geocoding and reversed geocoding, which allows also for address disambiguation.

Precise connection to the transport network. Many journey planners identify the two nodes of the network that are closest, respectively, to the origin and destination points of the requested trip, and then compute the shortest (best) route between them. But this approach may determine an unsatisfactory route, especially in dense urban areas, where many transit stops are available; the search algorithm could in this case consider as origin node a stop of a line that is not really useful to reach the destination node. To overcome this problem, our search algorithm implicitly connects the actual origin and destination points to all the nodes of the network that are closer than the maximum between: the distance between the closest node to the terminal point multiplied by a given coefficient; and a given threshold. For instance, a passenger whose home is 3 km away from the closest transit node will consider to begin his/her trip all the stops that are contained in the circle of radius 4.5 km centred at his/her origin, if the coefficient is 1.5.

Access and egress alternatives. The speed and cost of travelling along the implicit direct link from the origin point to a transit stop (access) or from a transit stop to a destination point (egress) can vary depending on the auxiliary transport mode utilized to this end: walking, biking, taxi, secondary transit lines, ect. These can be explicitly described in the dataset, so that the routing engine will provide the best combination of access stop, egress stop and route, together with the auxiliary transport modes, based on the user travelling preferences.

Waypoints. It is possible to introduce in the request an ordered set of intermediate points with the corresponding duration of the stops.

Fast algorithm. The routing engine is constituted by a shortest hyperpath algorithm with flexible bucket list. An A* like speed-up is also introduced, which allows to direct the exploration toward the destination. The computing time of the resulting procedure varies less than linearly with the number of nodes in the network. This unique features allow to process hundreds of path requests per second on large networks. Moreover, our search method is natively dynamic and thus allows to consider in one framework time-varying impedances as well as time-scheduled services.

Real time computation. Each path request sent by the clients is elaborated in real time through an independent run of the search algorithm. Moreover, since no pre-computation of the best routes is used, the system is capable of using the most recent available information about the road and transit network attributes, such as: traffic congestion, carrier delays, road closures, run cancellations. The software connectors between the engine application and the live database have to be developed for each specific project.

Dynamic link and line attribute. Given the network topology, the main attributes of road links and transit lines (speeds, frequencies, etc.) can be assumed to vary during each day-type, by defining proper time slices with constant values. As another option the travel times of road links can be assumed to vary linearly during each time slice.

Dynamic shortest paths. The routing algorithm can provide the dynamic shortest path, by taking into account in the optimization the estimated/forecasted performances that will occur at the actual time when the user will travel on each link of the network. This clearly requires to ask the user also for his/her departure time from the origin.

Multimedia route description. The optimal hyperpath provided by the routing engine is presented in the client browser in both textual and graphical formats, as a sequence of actions. To each text line describing an action corresponds a polyline on the map and viceversa.

Link by link statistics. The answer to the client that made the path request is provided via standard xml formats, reporting all travel cost components link by link.

Line offset on the map. The overlapping line segments composing the best hyperpath (which may be rather complex, in theory) are properly offsetted to ensure its readably by the user.

Attributes on the map. It is possible to highlight on the map the links characterized by given attributes, as in a web gis.

Hierarchical text list. To improve the accessibility of the solution, the list of actions can be expanded and grouped, following the hyperpath hierarchy.

From text to map and vice versa. Since textual actions and graphical segments are associated one to one, by clicking a given link on the map, the corresponding line in the text list is highlighted. Vice versa, by clicking on a given text line, the corresponding segment on the map is highlighted. This feature greatly enhances the navigation of the best route.

Comprehensive log. The system can provide a complete log of all the path requests and of all the route descriptions answered, together with a detailed monitoring of the algorithm performances.

OD matrix estimation. The complete historic set of paths requested to the system is a valuable information that can be used for different purposes. In particular, the requests can be spatially aggregated geo-referring origins and destinations to a traffic zone layer, thus providing a suitable estimation of an O-D trip matrix.

Multilanguage. Both the client interface and the route descriptions can be navigated in a effective multilanguage environment.

Smart phones. The system is accessed through the web, but smart-phone dedicated pages are provided to improve visualization and usability.

Traffic map with los. If data are available, a coloured map with real time or forecasted level of services and traffic flows will be provided as a background.

Traffic events. Thus functionality is provided only to the network managers. It allows to manually introduce traffic events, such as road closures and reductions of speeds. These events are taken into account in real-time when computing the optimal routing.

ARCHITECTURE

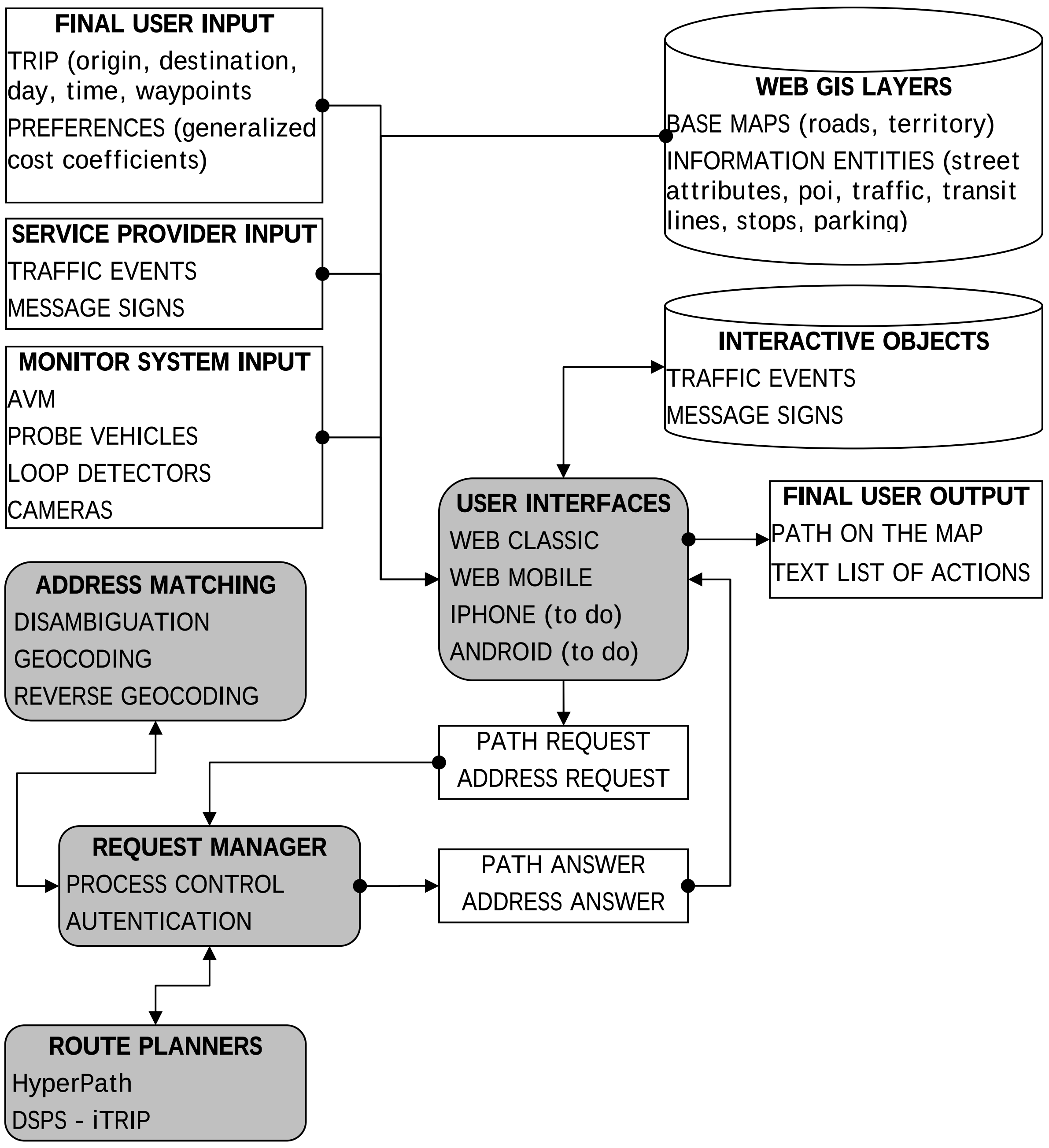
The architecture of the system has four main software modules.

User interfaces. This is the client component. Is developed in Javascript

Request manager. This is the pivot component server side. It is aimed at processing all the requests coming from the clients.

Address matching. This is a server side component, but can be also some third party service provider such as Google.

Route planners. This is the main application server side, where the algorithm runs.


FINAL USER INPUT
TRIP (origin, destination, day, time, waypoints
PREFERENCES (generalized cost coefficients)
SERVICE PROVIDER INPUT
TRAFFIC EVENTS
MESSAGE SIGNS
MONITOR SYSTEM INPUT
AVM
PROBE VEHICLES
LOOP DETECTORS
CAMERAS
WEB GIS LAYERS
BASE MAPS (roads, territory)
INFORMATION ENTITIES (street attributes, poi, traffic, transit lines, stops, parking)
INTERACTIVE OBJECTS
TRAFFIC EVENTS
MESSAGE SIGNS
USER INTERFACES
WEB CLASSIC
WEB MOBILE
IPHONE (to do)
ANDROID (to do)
FINAL USER OUTPUT
PATH ON THE MAP
TEXT LIST OF ACTIONS
ADDRESS MATCHING
DISAMBIGUATION
GEOCODING
REVERSE GEOCODING
PATH REQUEST
ADDRESS REQUEST
REQUEST MANAGER
PROCESS CONTROL
AUTENTICATION
PATH ANSWER
ADDRESS ANSWER
ROUTE PLANNERS
HyperPath
DSPS - iTRIP