

Hybridisation of Constraint Solving with an Ant Colony Algorithm for Vehicle On-Line Path Planning

Christophe Guettier¹, Francois Lucas¹ and Patrick Siarry²

¹SAGEM, 27, Rue Leblanc, 75012 Paris, France
{christophe.guettier, francois.lucas}@sagem.com

²Univ. of Paris XII Val-de-Marne, Sciences and Technology Faculty
61, av du G^{al} de Gaulle, 94010 Creteil, France
patrick.siarry@univ-paris12.fr

Abstract

This paper presents a hybrid solving method for vehicle path planning problems. As part of the vehicle system architecture (vetronic), planning is dynamic and has to be activated on-line which require response times to be compatible with mission execution. The proposed approach combines a complete method based on constraint solving techniques with an Ant Colony (ACO) metaheuristic. ACO is used to solve a relaxed problem as a pre-processing step. The hybridisation then relies on a probing technique that order variables according to a metric built on a distance information to the best solution found by ACO, allowing to guide a Branch&Bound search method. Various forms of strategies are compared and evaluated on real world scenarios. Preliminary results exhibit response times close to vehicle control requirements, on realistic problem instances.

Introduction

Mainly in space and defense domains, mission planning has always been a major challenge for the planning community, in terms of problem formulation, modelling, search techniques and evaluation. In critical mission systems for military vehicles, planning has been so far considered separated from navigation, in particular at the tactical level. However, modern operations take place in urban environments which involve versatile threats and require high tactical mobility.

For manned vehicle applications, the goal is to provide driver decision support functionalities, such as advising the best route to follow under specific mission constraints. To face environment uncertainty (e.g. obstacles, hostile threats), on-line planning algorithms must have execution times close to that of human reflexes. For unmanned vehicles (UAV, UGV, etc.), navigation plans must be updated whenever the mission objective changes, the environment evolves significantly or the expected amount of resources is not satisfactory. In addition, and according to the vehicle type and its on-board system (also called vetronic in the following), on-line planning must be compatible with mission tempo.

Much research has been carried out on mission planning. Generic planning formalisms (Long and Fox 2000)(Ghallab et al. 1998) have highlighted problem complexity, which

Copyright © 2009, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

can be tackled through domain-independent search methods and heuristics. These approaches may not match on-line planning system requirements, embedded in vehicles. Specific techniques can fulfil on-line requirements such as in (Meuleau and al 2008), but may not encompass the spectrum of operational constraints.

This paper focuses on Constrained Vehicle Planning (CVP) problems. A hybrid path planning method is presented to address dynamic mission management for both manned or unmanned vehicles. To match both on-line response time and multiple requirements, we combine a complete solving based on Constraint Programming (CP) and Branch&Bound strategy with an Ant Colony Optimisation (ACO) algorithm. ACO is a stochastic metaheuristic that adapt easily to path planning and is efficient for discrete and dynamic state space problems (Di Caro & al 2005). We use it in a pre-processing step to build a probe that guides a Branch&Bound solving. CP provides generic expressiveness and efficient solving techniques for global optimisation and constraint satisfaction. Also, the problem instances we consider are relatively small, since environment horizon under consideration is limited.

To our knowledge, our approach is new. Other work has been led on ACO-CP hybridisation in the literature (Solnon et al. 2008) in which a CP solver uses an ACO algorithm as heuristic and backtracking method. Nevertheless, it is an incomplete method. In our case, the ACO solution is only used to order variables but the solving is still complete.

Experimentations underline interesting performances on representative problems (mission, urban, or open environment). Memory consumption, computation load and execution time are compatible with the shape of problem instances as well as on-line requirements.

Vehicle Navigation

Constrained Path Planning

The CVP problem consists in finding a path from the current vehicle location to an objective waypoint. Intermediate mandatory waypoints can be imposed to fulfil secondary objectives. This problem can be transformed into a TSP, known as NP-complete : the core difficulty is then to find an optimal sequence of mandatory waypoints to visit. According to user experience, two kinds of plan optimisation are interest-

ing: minimising mission duration and maximizing mission safety. However, various other operational metrics can be imposed as hard constraints, and represented as distances, such as energy or capabilities. The optimisation criterion is minimizing the time to destination. In addition, only vehicle energy will be considered as a secondary metric.

Example

As an illustrative example, let us consider the following situation inspired from a real case (fig. 1). During mission execution, the planner has only partial awareness of its environment. The knowledge horizon is given by ground observability, provided by vehicle team, sensors or external observations. Yellow-filled circles and arrows represent waypoints and feasible paths between waypoints respectively. Transparent circles and dotted arrows represent waypoints and transitions that are situated beyond the observability horizon. One or more waypoints, as the bold waypoint on the figure, may be imposed along the vehicle route.

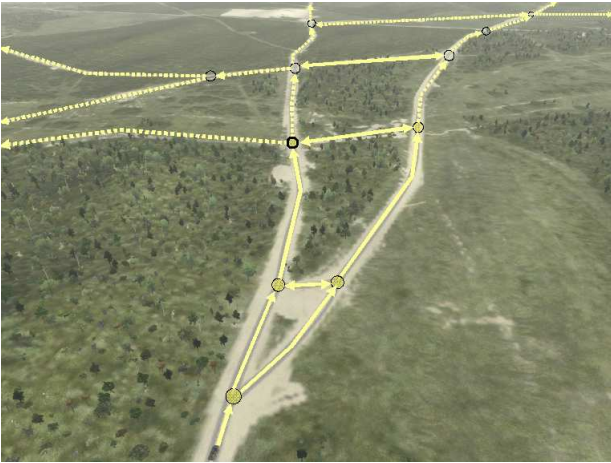


Figure 1: Example of alternative paths for a ground vehicle horizon, and corresponding to the line of sight of a cooperating UAV.

Environment horizon and response time

In our approach, on-line planning is solved over a limited horizon, from the current vehicle position. It corresponds to the terrain on which the vehicle vetronic has enough detailed information to characterise its trafficability. Therefore, the number of mandatory waypoints is relatively small. They correspond to short-term objectives or narrow manoeuvres executed by the vehicle. The following requirements drive response time of on-line planning:

- Computation time must be consistent with mission tempo, so that secondary objectives are reached.
- The generated plan must comply with vehicle control envelope. The control envelope on immediate feasible trajectories decreases when plan generation response time increases.

- In the case of manned vehicles, response time has to match pilot anticipation skills.
- In the case of unmanned vehicles, other processing (situation awareness, generation of flight control commands) must be done.

Note that if a plan cannot be quickly solved, the vehicle may stop if this is possible. This is an ultimate solution that is not satisfactory from an operational point of view. Finally, due to vetronic processing resources, computation load, memory usage and response time must be reduced as much as possible.

Hybrid solving approach

CP advantages combine a high level of expressiveness and powerful constraint solving techniques. It matches composite problems like CVP which requires formulation of different related models (Van Hentenryck et al. 1995). Following this approach, a flow $\{0, 1\}$ model of planning problems is proposed, which supports multiple distance metrics. This graph-based model of the terrain is very useful to represent tactical mobility (positions, progression axes, objectives), and vehicle abilities. CP also provides primitives such as Arc Consistency (AC) for constraint propagation, Branch and Bound (B&B) for optimisation and tree search (backtracking).

The ant colony search is a stochastic approach that combines heuristic search and learning in multiple cycles. It uses the same problem representation, but with a relaxed formulation. Only a single metric is considered and secondary objectives are modelled with a vertex parameter. Within a given cycle, a set of ants is deployed and finds some paths over the graph. A probabilistic law, depending on experience, is associated to a given ant in order to decide over alternative edges. Good quality solutions incrementally update the experience over several cycles. For a given cycle, the previous experience guides the search by attracting ants toward good path solutions. This technique allows the search to explore the state space despite any other implemented heuristic.

The hybridisation approach uses a static probing technique. The goal is to guide a complete strategy with the stochastic algorithm. The prober encapsulates the ACO search, which returns a probing solution to the relaxed problem. Instead of dynamic probing with tentative values such as in (El Sakkout & Wallace 2000), this search strategy uses a static prober which orders problem variables to explore according to the relaxed solution properties. Then, the solving follows the CP search strategy, combining B&B and AC.

Constrained Vehicle Planning Formulation

The terrain is represented as an undirected graph structure (see Fig. 1), where edges define progression axes and vertices tactical positions (or locations). Vertices also represent primary and secondary objectives. Other constraints can impose vertices or edges to be excluded or included in the vehicle plan. Lastly, operational metrics (protection, vulnerability, capacity) can also be associated to edges. The input specification can be expressed using terrain structure, initial

conditions, mission objectives and vehicle capabilities. The following elements are known off-line and characterise this input specification:

- **Initial conditions:** The starting location and resources initially available.
- **Objectives:** Some of the locations can correspond to secondary or primary objectives.

Basic constraints

The space of possible plans is represented as a directed graph $G(X, U)$ where the set of edges U represents possible progression axes and the set of vertices X possible position (or navigation) locations¹. A vehicle starts from vertex $start$ and must reach its objective at vertex end . A path is defined by the set of positive flows. A set of variables $\varphi_u \in \{0, 1\}$ models a possible path from $start$ to end , where the edge u belongs to the path if and only if decision variable φ_u is instantiated to 1.

From an initial position to a final one, path consistency is asserted by the following constraints, where $\omega^+(x) \subset U$ and $\omega^-(x) \subset U$ are outgoing and incoming edges from vertex x , respectively.

$$\sum_{u \in \omega^+(start)} \varphi_u = 1, \quad \sum_{u \in \omega^-(end)} \varphi_u = 1, \quad (1)$$

$$\forall x \in X \setminus \{start, end\}, \quad \sum_{u \in \omega^+(x)} \varphi_u = \sum_{u \in \omega^-(x)} \varphi_u \leq 1 \quad (2)$$

Nodes $start$ and end respectively represent current position and primary objective for the vehicle. Equation (2) ensures path connectivity and unicity while equation (1) imposes limit conditions for the extremities of the path. This constraint gives a linear chain alternating positions and mobility actions (along progression axes) along the graph.

Capability metrics

Assuming a given date D_x associated with a position (e.g. vertex) x , we formulate path length formulation (3) often considered in Operation Research (OR) (Gondran and Minoux 1995). Variable D_x expresses the time at which the vehicle reaches position x (see example in figure 2). Assuming that constants $d_{(x',x)}$ represent the time taken to perform a movement from location x' to x , we have:

$$\forall x \in X, D_x = \sum_{(x',x) \in \omega^-(x)} \varphi_{(x',x)} (d_{(x',x)} + D_{x'}) \quad (3)$$

Constants $d_{(x,x')}$ are critical decision variables in the problem and make constraints (3) non linear by terms $D_{x'}$. Finally, the mission schedule can be represented as $\Delta = \{(x, D_x) \mid x \in X, D_x > 0\}$.

An equivalent constraint-based formulation is also used for other mission metrics (Fig. 2), such as energy or capacity.

¹In the remaining of the paper, a vertex is denoted by x , while an edge can be denoted either by u or by (x, x') .

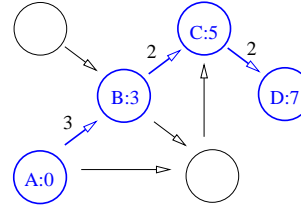


Figure 2: Illustrating a path with pass-by dates over a graph of locations and progression axes

Graph in fig. is a spatial representation of possible moves (edges) and positions (nodes). Moves, that correspond to the set of positive values $\Phi = \{(A, B), (B, C), (C, D)\}$, are represented with blue arrows. Assuming a timing metric (edge values are speeds). Other operational metrics, such as protection, vulnerability, available energy and security are similarly formulated in different experiments.

Hybrid Search

The solving strategies focus on mission duration optimisation, that is minimising the time to destination. This date corresponds to one of the variable set $\{D_{end}\}$. The position end is the primary objective of the vehicle. Decision variables are path variables $\{\varphi_x\}$, timing variables $\{D_u\}$.

Reference algorithm

The basic algorithm is a “generate and test” approach that is described only for complexity analysis purpose. Between any couple of secondary objectives, a shortest path is pre-computed. The algorithm then builds a quotient graph where the set of nodes includes all secondary objectives in addition to start and end ones. Quotient graph edges result from the precomputed shortest path, valued with the distance. Here the reference algorithm can only be applied for the relaxed problem (without energy or capability metrics). Algorithm complexity is $O(n!)$.

CP search strategy

All problem formulations and search strategies have been implemented in the $CLP(FD)$ ² SICStus prolog library. Basic search strategy makes use of B&B *minimise* predicate and $CLP(FD)$ constraint AC propagation algorithm. The B&B iterates over an arbitrary order of variables labeling. When a variable choice is done, AC propagates domain variables until a fixed point is reached.

Shortest path hybridisation

Designing a strategy consists in finding the right variables ordering and value filtering. The idea is to use the proper to statically order problem variables, as a preprocessing. Instead of using this solution as an initial tentative value, the proper estimates a distance between any problem variable and the probing solution. The search strategy then defines both variable and value ordering according to the resulting distance set. The technique does not suppress any choices points, such that the solving remains complete.

²Constraint Logic Programming, using Finite Domain as algebraic interpretation.)

Ant path hybridisation

Instead of considering a blind shortest path as prober, the proposed algorithm implements an ACO search that uses a similar model of the environment. As introduced before, the method deploys a set of ants over search cycles (the number of ants and the number of cycles being currently defined statically) and learns. At the end of each search, ants provide a set of paths, ordered according to their quality. Best ones (on the model of *Rank-based Ant System* (Bullnheimer & al 1999)) are selected to update experience by reinforcing weights associated to their edges. The experience enables search guidance towards interesting areas (where good solutions were found). Within an ant search, the choice of the next waypoint is given by the following probabilistic equation. For an ant k currently at vertex x , the probability for choosing a vertex x' as its next waypoint is given by:

$$\forall(x, x') \in X^2, x' \in \omega^+(x), P_{(x, x')}(k) = \frac{\tau_{x, x'}^\alpha \eta_{x'}^\beta}{\sum_{l \in \omega^+(x)} \tau_{x, l}^\alpha \eta_l^\beta} \quad (4)$$

where $P_{(x, x')}$ is a probability and thus belongs to $[0, 1]$. The $\eta_{x'}$ parameter represents the search heuristic (which is simply the inverse of the minimal distance to go from vertex j to the next objective in this implementation). This parameter tends to choose the closest vertices to the objective. The $\tau_{(x, x')}$ parameter represents the edge weight to go from vertex x to vertex x' . This represents the experience acquired during previous search cycles, which tends to choose edges that belong to known good solutions. Parameters α and β are used for calibration and balance the importance between τ and η parameters.

The hybridisation schema of the ACO algorithm is similar to the shortest path one.

Discussion

There is no universal rule to parameterize the ACO algorithm. It depends on the problem, essentially in terms of graph size and connectivity. The choice of α and β can be decisive if there is a high risk for the search to follow a dead-end path. In this case, by privileging the η term (thus by setting β higher than α), the search will have more chance to fail (or to find bad solutions) if it takes a path towards the objective but which does not lead to it. An analogy can be made with A^* algorithm, whose worst case is a labyrinth in which the only way to reach the objective is opposite to the location's direction. In the other cases, it can be judicious to take more consideration to η that can fastly lead to good solutions. The N parameter depends on the size of the problem. The larger the problem is, the more important the ant population (represented by N) should be, as the number of possible solutions becomes high. The C parameter (number of search cycles, analogous to generations in genetic algorithms) more particularly depends on learning mechanisms. As we know, the quality of the search is a compromise between state space exploration and convergence speed. According to the chosen strategy, C should be low if a fast convergence is wished (the reinforcing edge parameters should

then evolve fastly), or high in the contrary (and reinforcing should be mild to maintain alternative paths).

In the current version of our ACO algorithm, parameter values were empirically fixed as follows: $\alpha = 0.6$; $\beta = 1$; $\eta = 0.6$; $C = 6$; $N = 6$; $\rho = 0.08$ (ρ is the pheromone evaporation factor, used to decrease edges attractiveness over time).

Preliminary Results

To evaluate the efficiency of the hybrid algorithm, we compared the performance results with two other search methods : a simple constraint solving algorithm based on arbitrary variable ordering, and a hybrid constraint solving - shortest path algorithm.

Experiments have been run on a dual core CPU, at 2.53GHz with 2Gbytes of memory.

Problem instances To illustrate the approach, experiments on three benchmarks are presented. They are representative of vehicle planning for modern peace keeping missions, both in urban and open environments. The following table gives an idea of problem complexity.

Problems	Bench1	Bench2	Bench3
Environment	urban	urban	open
Vertices	23	22	22
Edges	76	74	68
Variables	723	654	702
Constraints	1944	1750	1886

For each benchmark, fifty distinct problem instances are generated, organised on five series. For a given serie, ten instances with starting and ending nodes are chosen on the graph diameter. For each serie, a fixed set waypoints is imposed. Over the five series the difficulty increases from one imposed waypoint to five ones (see example fig. 3).

Results Response times are given in figure 4 (the y-axis) for the three benchmarks, over the five series (x-axis). The measurement considers the search strategy response time, including the whole probing preprocessing.

The y-axis shows response time range for a given serie. The number of imposed waypoints does not affect response time for hybrid search strategies, compared to the reference algorithm complexity. In contrast, only benchmark 1 is a problem for the basic CP search, when increasing difficulty. Except on benchmark 3, the basic strategy delivers poor results, and even exceeds the ten seconds limit twice on the first benchmark. This underlines the efficiency of variable ordering approach. In twelve experiments over fifteen, ACO algorithm dominates the shortest path guided strategy. However, both hybridisation approaches have response times of a similar order of magnitude.

The total number of backtracks over the 10 runs is presented in figure 5 (y-axis) for each serie (x-axis). The number of backtracks is measured during the B&B search execution. In general, the number of backtracks reflects response times. Also, computation time is not spent on optimisation iterations but on backtracking and variables labeling. For both hybrid strategies, variable ordering guides the search

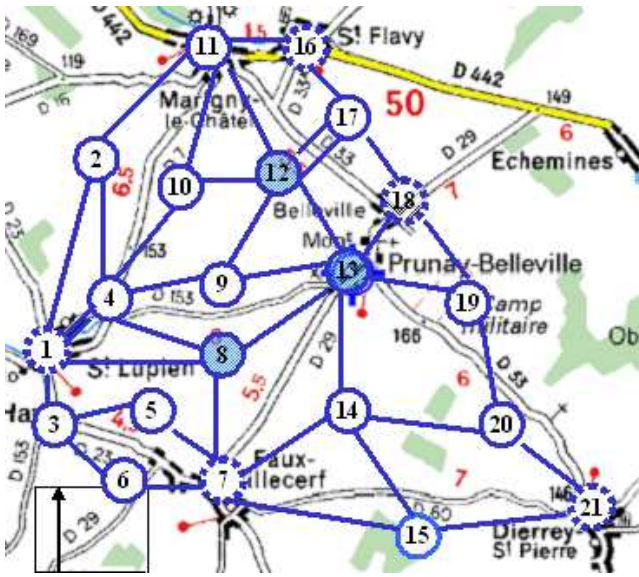


Figure 3: A serie of 3 imposed waypoints (colored circles) on an open environment benchmark. Dashed circles represent possible starting or ending nodes.

significantly so that it reduces the amount of backtracks. In fact, hybrid algorithms find the optimal solutions in few B&B iterations, with minimum backtracks. When comparing these two, ACO search gives better results (a lower number of backtracks), as metaheuristic outputs provide better strategy guidance. On series 2, 3 and 5 of benchmark 3, both hybrid strategies counterperform, compared to the pure CP search strategies. This suggests that the order (formulated by an operational expert, which generally follows a temporal order) is not so arbitrary! This is the case even for the hybrid ACO strategy, in spite of a lower number of backtracks. In these three series, computation time is spent on B&B optimisation for the hybrid ACO strategy while computation is lost in backtracking for the shortest path hybrid one.

The following table compares overhead computation for both shortest path and ACO algorithms. The latter provides acceptable execution time, and is worth the computation savings during the CP search.

Search cycle number and ants population size have been found acceptable for the problem. When these parameters are badly chosen, the metaheuristic may not find a solution. A naive approach would be to increase parameters as much as possible, but it can seriously impact solving time. In fact, these parameters choices represent a compromise between solving time and chances to find a (good) solution.

Overheads	Bench1	Bench2	Bench3
Shortest Path (ms)	[0, 30]	[0, 31]	[0, 16]
Ant Colony (ms)	[31, 110]	[31, 94]	[47, 109]

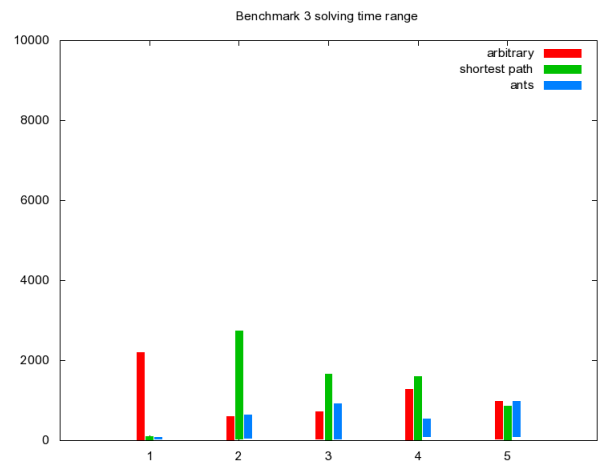
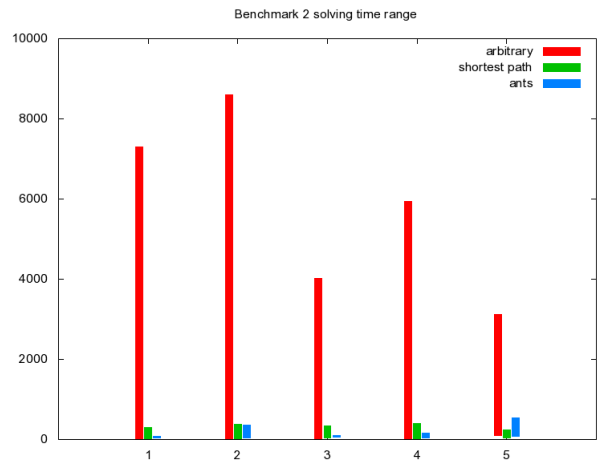
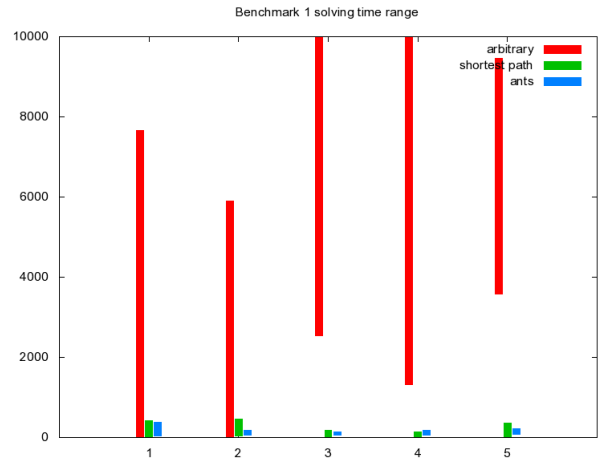


Figure 4: Minimum and maximum response times to reach optimal solution for the three benchmarks, with series of one to five number of imposed waypoints.

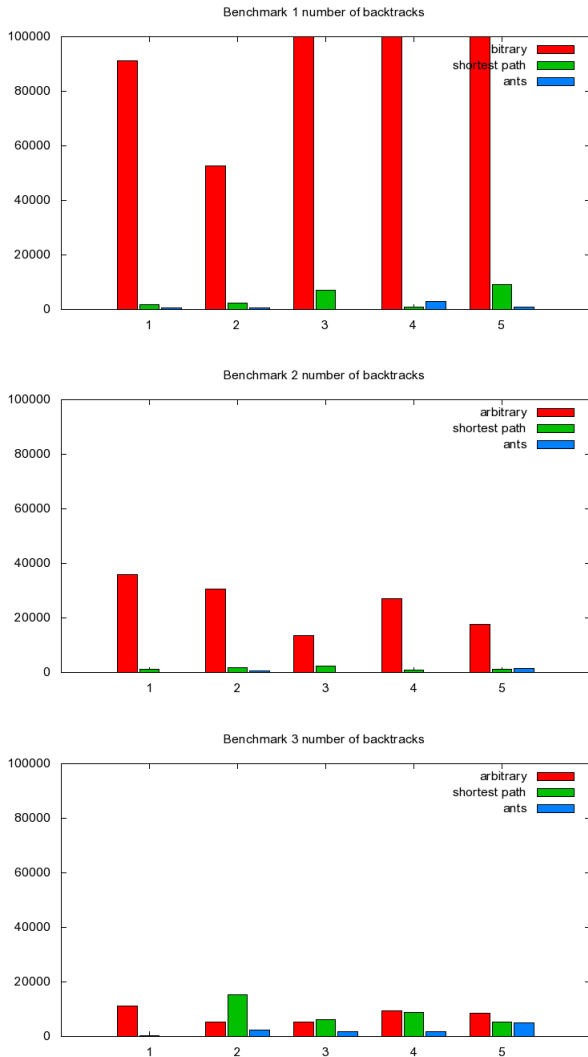


Figure 5: Total number of backtracks needed to find the optimal solution for the three benchmarks. It is summed over series of one to five number of imposed waypoints.

State of the art

Different techniques have been shown to be useful to tackle on-line planning problems, including theoretical formalisms, generic or specific heuristics, constraint solving, and local search techniques.

- **Generic planners:** Generic planning techniques can be applied to solve such problems. In (Long and Fox 2000), transportation problem classes are proposed, for which preprocessing and dedicated heuristics can be introduced to specialise generic search algorithms.
- **Domain specific planners:** Much planning research has been done for both military or civilian purposes, relying on specific planning frameworks such as Hierarchical Task Network (HTN)(Goldman et al. 2000).
- **Planning with constraint solving:** This is the case in Ix-TeT (Laborie and Ghallab 1995) and HSTS. In Reactive Model-based Programming Language (RMPL) (Kim & al 2001), an evolution of CC languages, the same paradigm is used to dynamically constrain planning representations of one or more remote agents.
- **In many respects, CVP can be tackled with operation research algorithms, based on flow models.** Basically, the CVP problems can be relaxed as a Traveling Salesman Problem, for which numerous algorithms have been proposed (see (Gondran and Minoux 1995) for example). Local searches have also been widely used for TSP problems. In particular, new metaheuristics such as ACO algorithms can be efficiently applied (Aarst and Lenstra 1997) to TSP.
- **Much work addresses dynamic on-line planning, including computer games and exploration vehicles (like NASA's Mars Rover).** Two efficient kinds of algorithms are particularly widespread : real-time heuristic search methods (like LRTA* (Korf 1990) or RTAA* (Koenig and Likhachev 2006)) and incremental heuristic search methods (like D* Lite (Koenig and Likhachev 2002)). The first one only consider a local environment subset to solve, which is updated over time, to limit the problem size. The second approach consider the whole environment but reuses data from previous searches to gain execution time and avoid dead-ends.

Most of constraint programming frameworks are useful to design hybrid search techniques, by integrating OR and Linear Programming algorithms (Ajili and Wallace 2003). However, only a few ones such as (Knight et al. 01), have explored on-line planning requirements.

Conclusion

In this paper, we proposed a new hybrid ACO - CP algorithm to tackle vehicle path planning, whose response times must fit on-line requirements. The CP approach allows higher constraint expressiveness and global solving abilities, while the ACO algorithm is used as a probe to order search variables. The experimentations led on realistic examples clearly showed the interest of variables ordering for search guidance, resulting in a significant reduction of

response time. They also revealed better performance of the ACO algorithm over the shortest path probing, both in terms of solving time and total number of backtracks. However, calibration of ACO algorithms must be tuned according to problem size and structure. An interesting way to improve our approach would be to dynamically parameterise the ACO metaheuristic to fit problem and vetronic requirements. In this preliminary approach, starting location and resource availability are off-line parameters of our solver. A transition to dynamic aspect can be introduced by adding a sliding environment horizon, updating the state space as well as the vehicle state (current position, objectives and remaining resources). Here, the interest of ACO algorithms is to use edge reinforcement of previous searches in new computations. A comparison of this dynamic ACO with other well-known heuristic methods like LRTA* or D* Lite will also be investigated.

References

- P. Albert, M. Khichane, C. Solnon : Integration of ACO in a Constraint Programming Language. In 6th International Conference on Ant Colony Optimization and Swarm Intelligence (ANTS), Bruxelles. pp. 84-95. LNCS 5217. Springer.
- Aarst,E. and Lenstra,J.K. 1997. Local Search in Combinatorial Optimisation, McGraw Hill, Chichester, UK, 1997.
- Ajili,F.; Wallace,M. 2003. Constraint and Integer Programming: Toward a Unified Methodology. In Chapter 6: Hybrid Problem Solving in ECLiPSe. Kluwer Academic Publishers, 2003.
- Knight R.; Rabideau G.; Chien S.; Engelhardt B. and Sherwood R. Casper: Space Exploration through Continuous Planning. IEEE Intelligent Systems, vol 16, P.70, 2001.
- El Sakkout,H. and Wallace M. Probe Backtrack Search for Minimal Perturbations in Dynamic Scheduling. Constraints Journal, Vol. 5, 2000.
- Ghallab,M.; Howe,A.; Knowblock,C.; Mac Dermott,D.; Ram,A.; Veloso,M.; Weld,D.; and Wilkins,D. 1998. PDDL- The Planning Domain Definition Language. Technical Report, CVC TR-98-003/DCS TR-1165.
- Gondran,M. and Minoux,M.1995. *Graphes et Algorithmes*, ed. Eyrolles, Paris.
- Goldman,R.P.; Haigh,K.Z.; Musliner,D.J.; and Pelican,M. 2000. MACBeth: A Multi-Agent Constraint-Based Planner. Proceedings of the AAI Workshop on Constraints and AI Planning. Menlo Park, Calif.: AAI Press.
- Laborie,P. and Ghallab,M. 1995. Planning with sharable resource constraints. Proceedings of IJCAI'95. Menlo Park, Calif.: International Joint Conference on Artificial Intelligence, Inc.
- Long, D. and Fox, M. 2000. Automatic Synthesis and use of Generic Types in Planning. Proceedings of AAI 2000. Menlo Park, Calif.: AAI Press.
- Meuleau,N.; Plaunt,C. and Smith D. 2008. Emergency Landing Planning for Damaged Aircraft. In SPA Workshop, International Conference on Automated Planning and Scheduling.
- Van Hentenryck,P.; Saraswat,V and Deville,Y. 1995. Design, Implementation and Evaluation of the Constraint Language CC(FD), In Constraint Programming: Basics and Trends, A. Podelski Ed., Springer-Verlag.
- Kim,P.; Williams, B and Abramson, M. 2001. Executing Reactive, Model-based Programs through Graph-based Temporal Planning. Proceedings of the International Joint Conference on Artificial Intelligence, Seattle, WA.
- Bullnheimer B., Hartl R., Strauss C., A new rank-based version of the ant system : a computational study , Central European Journal for Operations Research and Economics, vol. 7, n1, p. 2538, 1999.
- Di Caro G. A., Ducatelle F., Gambardella L., AntHocNet : an adaptive Nature inspired algorithm for routing in mobile ad hoc networks , European Transactions on Telecommunications (ETT), vol. 16, n5, 2005.
- R. Korf : Real-Time Heuristic Search, Artificial Intelligence 42 (2-3), pages 189-211, 1990.
- S. Koenig, M. Likhachev : Real-Time Adaptive A*, in Proceedings of the AAMAS, pages 281-288, 2006.
- S. Koenig, M. Likhachev : D* Lite, in Proceedings of the AAI, pages 476-483, 2002.