JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 32, No. 2, March–April 2009

Path-Planning with Avoidance Using Nonlinear Branch-and-Bound Optimization

Alison Eele* and Arthur Richards†
University of Bristol, Bristol, England BS8 1TR, United Kingdom

DOI: 10.2514/1.40034

This paper describes a novel method for finding optimal trajectories for a vehicle constrained to avoid fixed obstacles. The key property of the method is that it provides globally optimal solutions while retaining the full nonlinear dynamics model. Applications for the method include guidance of unmanned aerial vehicles, air traffic control, and robot path planning. The core concept is the *direct* application of branch-and-bound optimization to find guaranteed, globally optimal solutions to nonconvex problems. The method tailors the branch-and-bound approach specifically for avoidance problems by exploiting two new ideas: first, using a geometric branching strategy based on the decision between passing an obstacle clockwise or counterclockwise; and second, solving the resulting subproblems by constructing simple solutions on each chosen "side" and using them to initialize an interior-point optimization. The algorithm is refined by comparing nine geometric branching strategies. The solution time of the method depends on the choice of branching strategy, which determines how the solution tree is explored. A good strategy is one requiring fewer tree branches to be enumerated before the global optimal is found. The best of these branching strategies has been compared with an existing mixed-integer linear programming approach and demonstrated a significant improvement on mixed-integer linear programming solve times.

Nomenclature

$a_{\max}$	=	maximum acceleration
N	=	total number of obstacles
ncol	=	number of collocation points used to discretize a path
P_a, P_c	=	"diverted" paths, counterclockwise and clockwise, respectively, around a particular obstacle
P_{inc}	=	incumbent path
P_s	=	optimized path for a particular subproblem
R_p	=	radius of obstacle p
$(r_{x,p}^{\text{obs}}, r_{y,p}^{\text{obs}})$	=	position of center of obstacle p
$(r_x(t), r_y(t))$	=	position at time t
$(r_{x0}, r_{y0}, v_{x0}, v_{y0})$	=	initial position and velocity
$(r_{x,m}, r_{y,m})$	=	mth collocation point
(r_{xT}, r_{yT})	=	target position
t_f	=	final time
t_0	=	start time
$v_{\max}$, $v_{\min}$	=	maximum and minimum velocities
$(v_x(t), v_y(t))$	=	velocity at time t

I. Introduction

THIS paper develops a novel algorithm guaranteed to find globally optimal solutions to two-dimensional nonlinear vehicle trajectory design problems subject to avoidance constraints. These problems are important for the control of autonomous uninhabited aerial vehicles (UAVs) [1] and air traffic management [2]. They are also very complex, known to be in the class of nondeterministic-polynomial time complete problems [3,4].

Many approaches have been investigated for solving these problems. All involve some kind of simplification, aiming to capture certain key elements of the task in a form suitable for practical computation. Methods based on nonlinear programming (NLP) can incorporate nonlinear dynamics models but may converge to local minima [5–12]. This is a particular problem for obstacle avoidance constraints, where locally optimal routes exist on either side of an obstacle, and we wish to decide which route to take. Other methods have explicitly addressed the discrete decisions. Randomized searches [13–15] rapidly find feasible paths through fields of obstacles, retaining accurate dynamics models but having no guarantee of optimality. Graph-based methods such as Voronoi diagrams [9,16] or visibility graphs [17] approximate the trajectories as joined line segments. The mixed-integer linear programming (MILP) [18] approach uses *indirect* branch-and-bound optimization, reformulating the problem in linearized form and using powerful commercial software to solve the MILP problem. A heuristic method using MILP to initialize NLP has also been proposed [19,20]. Both the graph search and MILP approaches guarantee globally optimal solutions, but to approximated problems without the original nonlinear dynamics. As far as the authors are aware, the new method developed in this paper is the first to combine global optimization with nonlinear vehicle dynamics.

The new algorithm employs branch-and-bound optimization [21] to explicitly consider the discrete decisions involved. This builds on the success of the MILP approach [18], which indirectly uses branch and bound within the commercial solver. The branch-and-bound method cannot improve the overall computational complexity of the problem, which remains exponential in the number of obstacles and time steps, however, it can act as a heuristic to speed up results. As in most other work in this field, the vehicle is assumed to move in two dimensions, with the altitude fixed by other constraints such as airspace layers or operational requirements. Although the method can be used to optimize for a variety of objectives, such as risk or fuel consumption, this paper will consider finding the shortest-time path throughout.

There are two elements to a generic branch-and-bound method:

1) Branching: the problem is divided into subproblems by partitioning the search space. Each subproblem is further divided in the same way, and the algorithm proceeds by searching a "tree" of subproblems, hence "branching."

2) Bounding: a lower bound on the optimal cost of each subproblem is found by solving a relaxed, simpler form of that

Presented as Paper 6793 at the AIAA Guidance, Navigation and Control Conference and Exhibit 2007, Hilton Head, SC, 20–23 August 2007; received 25 July 2008; revision received 19 November 2008; accepted for publication 20 November 2008.

*Ph.D. Candidate, Department of Aerospace Engineering, Queens Building, University Walk; Alison.Eele@bristol.ac.uk. Student Member AIAA.

†Lecturer, Department of Aerospace Engineering, Queens Building, University Walk; Arthur.Richards@bristol.ac.uk. Member AIAA.

subproblem. These bounds are then used to identify if a branch requires further subdivision. If a) the subproblem is infeasible, b) the solution to the subproblem is a feasible solution of the complete problem, or c) the bound from the subproblem solution is worse than the cost of the best feasible solution found so far, then the branch is said to be "fathomed" and no further branching is necessary.

The following examples illustrate the direct mapping of branch and bound to the avoidance problem:

1) Branching: the problem of avoiding obstacles $1 \ldots M$ can be divided into two subproblems: a) the problem of passing clockwise around obstacle 1 and avoiding obstacles $2 \ldots M$, or b) the problem of passing counterclockwise around obstacle 1 and avoiding obstacles $2 \ldots M$.

2) Bounding: the shortest path passing clockwise (or counterclockwise) around obstacle 1 and avoiding obstacles $2 \ldots M$ must be longer than the shortest path passing clockwise (or counterclockwise, respectively) around obstacle 1 and ignoring obstacles $2 \ldots M$.

The global optimality of any solution obtained from a branch-and-bound method is guaranteed, assuming that the globally optimal solution to each evaluated subproblem is found [22]. In the avoidance case, this corresponds to finding the best path passing on a given side of a set of active obstacles and ignoring all others. Because this subproblem no longer involves a choice of side, it can be solved by nonlinear programming. This leads to the idea demonstrated in this paper: although it is difficult to express the concept of one side or another in terms of an explicit constraint, it can be achieved by initializing the search on the appropriate side and employing a primal-dual optimization method which is then checked against the desired path. Because each choice of side corresponds to a disconnected class of paths [4], in the unlikely case of a primal-dual optimization jumping across obstacles into a different disconnected path class, it is possible to detect and reject the solution. Therefore, effectively, the method ensures that the nonlinear programming path optimization is constrained to the same class of paths as the initialization.

The speed of reaching the globally optimal solution by the branch-and-bound method is affected both by the complexity of the problem and by the branching strategy, that is, the order in which the tree of solutions is explored. Many branching strategies have been investigated since the inception of the branch-and-bound approach. Two of the most popular are depth-first search [23–25] and best-bound search [26–28]. Depth-first seeks to select the newest subproblem created for branching and has been observed to require memory that is a linear function of the problem size [29]. Meanwhile, best-bound chooses the subproblem with the lowest bound on objective function of the parental subproblem to be branched on. This, however, takes no account of any other parameters than the objective function. Pseudocosts or estimations for linear cases were then proposed to include parameters such as the quality of a solution into the best-bound approach [30,31]. Breadth-first searching employs a first-in/first-out approach to the nodes, though it remains less popular than depth-first and best-bound due to the high number of subproblems required to be solved [32]. In summary, there is no one perfect generic branching strategy which will guarantee the minimal solve time on any proposed problem and the best branching strategy is highly problem specific [33]. Therefore, this paper includes experimental investigation of branching strategies.

The paper is organized as follows. Section II describes the problem statement. Section III describes the new branch-and-bound optimization algorithm and demonstrates it graphically. Section IV details the nine branching strategies. Section V presents illustrative examples of results found using the new algorithm. Finally, Sec. VI presents conclusions.

II. Problem Statement

This paper considers the problem of finding the shortest path for a vehicle modeled as a "Dubins-like" car, that is, moving in two dimensions with limited rate of turn, though with bounded speed. The velocity and positioning relationships are linked by the following constraints:

$$v_x(t) = \frac{\mathrm{d}}{\mathrm{d}t} r_x(t) \tag{1a}$$

$$v_y(t) = \frac{\mathrm{d}}{\mathrm{d}t} r_y(t) \tag{1b}$$

Speed is constrained between both minimum and maximum bounds,

$$v_{\min}^2 \leq v_x^2(t) + v_y^2(t) \leq v_{\max}^2 \tag{2}$$

The acceleration is also bounded, effectively restricting the turning rate:

$$-a_{\max} \leq \frac{\mathrm{d}}{\mathrm{d}t} v_x(t) \leq a_{\max} \tag{3a}$$

$$-a_{\max} \leq \frac{\mathrm{d}}{\mathrm{d}t} v_y(t) \leq a_{\max} \tag{3b}$$

The initial state is completely specified by the following constraints:

$$r_x(t_0) = r_{x0} \tag{4a}$$

$$r_y(t_0) = r_{y0} \tag{4b}$$

$$v_x(t_0) = v_{x0} \tag{4c}$$

$$v_y(t_0) = v_{y0} \tag{4d}$$

and the terminal constraint is that the vehicle should reach a specified target regardless of velocities at time t_f:

$$r_x(t_f) = r_{xT} \tag{5a}$$

$$r_y(t_f) = r_{yT} \tag{5b}$$

Finally, the obstacle avoidance is expressed as a minimum distance from each obstacle center. The method has been demonstrated throughout with the use of circular obstacles, however, can be converted to arbitrary obstacles by approximation as unions of circular obstacles:

$$\left[r_x(t) - r_{x,p}^{\text{obs}}\right]^2 + \left[r_y(t) - r_{y,p}^{\text{obs}}\right]^2 \geq R_p^2 \quad \forall\ t, \quad \forall\ p \in \{1, \ldots, N\} \tag{6}$$

The final time t_f is constrained such that

$$t_f \geq 0 \tag{7}$$

The objective is to find the minimum time path where the elapsed time $t_f - t_0$ is minimum:

$$J^* = \min t_f - t_0 \tag{8}$$

This paper assumes that the objective is the shortest-time path without loss of generality. The cost function is independent of the branch-and-bound method and can be substituted to solve a variety of problems such as those of minimum fuel use and minimum risk [27]. The dynamics are discretized using direct global collocation with Gaussian Radau points for constraint enforcement [34,35]. The position and velocity profiles are parameterized by

$$r_x(t) = \sum_{m=1}^{\text{ncol}} \psi_m\left(\frac{t - t_0}{t_f - t_0}\right) r_{x,m} \tag{9a}$$

Algorithm 1 Branch-and-bound path planning

1. $P_{\text{inc}} \leftarrow$ NULL
2. Set list of active subproblems to $(\emptyset, P_0)$ where path P_0 connects the start to the goal in a straight line
3. **while** list of active subproblems is not empty
4. **do** select an subproblem (A, P_i) from the list of active subproblems and remove it from that list
5. starting with path P_i, search for feasible path P_d, avoiding active obstacles A
6. **if** suitable P_d exists
7. **then** search for shortest feasible path P_s, avoiding active obstacles A
8. **if** path P_s intersects inactive obstacles $\{1, \ldots, M\} \setminus A$ and path P_s is shorter than P_{inc}
9. **then** select an obstacle p_b (the "branching obstacle") intersected by path P_s
10. construct path P_a (not necessarily dynamically feasible) by diverting P_s counterclockwise around obstacle p_b and add $(A \cup \{p_b\}, P_a)$ to the list of active subproblems
11. construct path P_c (not necessarily dynamically feasible) by diverting P_s clockwise around obstacle p_b and add $(A \cup \{p_b\}, P_c)$ to the list of active subproblems
12. **else, if** path P_s is shorter than P_{inc}
13. **then** $P_{\text{inc}} \leftarrow P_s$
14. **return** P_{inc}

$$r_y(t) = \sum_{m=1}^{\text{ncol}} \psi_m\left(\frac{t - t_0}{t_f - t_0}\right) r_{y,m} \tag{9b}$$

$$v_x(t) = \sum_{m=1}^{\text{ncol}} \psi_m\left(\frac{t - t_0}{t_f - t_0}\right) v_{x,m} \tag{9c}$$

$$v_y(t) = \sum_{m=1}^{\text{ncol}} \psi_m\left(\frac{t - t_0}{t_f - t_0}\right) v_{y,m} \tag{9d}$$

where $r_{x,m}$, $r_{y,m}$, $v_{x,m}$, and $v_{y,m}$ represent the values of the associated variables at collocation point m, and ψ_m is the mth Lagrange polynomial of the order ncol satisfying

$$\psi_m(\tau_r) = \delta_{m,r} \tag{10}$$

where τ_r is the rth collocation point and $\delta_{m,r}$ is the Kronecker delta. Thus, the path optimization is approximated by the finite dimensional optimization with decision variable $P = \{r_{x,1} \ldots r_{x,\text{ncol}}, r_{y,1} \ldots r_{y,\text{ncol}}\}$. The corresponding conversion of the constraints is familiar and is covered in detail in [34,35] for example.

III. Obstacle Branch-and-Bound Planning Algorithm

This section describes the new algorithm mathematically. A graphical illustration of the steps for a particular example is shown in Sec. III.D.

Define a *subproblem* (A, P) consisting of a set of active obstacles $A \subseteq \{1, \ldots, M\}$ and an initial diverted path P (represented by $\{r_{x,m}, r_{y,m}\}$ for all $m = 1 \ldots \text{ncol}$), which satisfies the initial and terminal constraints and the avoidance constraints for the active obstacles A but is not necessarily dynamically feasible. P_{inc} represents the incumbent path, that is, the "best" feasible path found so far, where best is defined as the shortest-time path in this paper.

The branching, or partitioning of the search space, is performed implicitly by the use of the diverted paths P_a and P_c as the initial solutions for subsequent problems and by using policed primal-dual interior-point optimization methods for each search in steps 5 and 7. The policing of the optimization methods enforces that paths P_d and then P_s must respect the "clockwise or counterclockwise" decisions passed down to them in the diverted path or be rejected.

The branch-and-bound method proposed does not assume any particular form of vehicle dynamics and thus the dynamics presented in Sec. II can be changed independently of altering the branch-and-bound method [36]. Under the assumption that each path P_s is the optimal solution to its corresponding subproblem, the properties of the generic branch-and-bound algorithm hold [22] and, at termination, P_{inc} is the globally optimal path. This is reasonable, as careful initialization (Secs. III.A and III.C) and checking of solutions has been used to significantly reduce the number of encounters with local minima. Validation of this approach is provided in the form of path-length performance comparisons with existing methods in Sec. V.C. The following subsections describe the key intermediate steps in detail.

A. Step 5: Search for Dynamically Feasible Path

Each subproblem is defined by an initial path which has been diverted around the active obstacles. This path is unlikely to be dynamically feasible, although some care is taken to at least have a good attempt, as discussed in Sec. III.C. The preliminary search for a dynamically feasible path in step 5 has two purposes: first, to generate such a path if one exists, and second, to identify cases where no such path exists, thereby quickly pruning out infeasible problems.

Before attempting to search for a dynamically feasible path, the method must first determine if the suggested path is logically feasible considering the potential for overlapping obstacles, that is, the combination of prescribed clockwise and counterclockwise diversions around potentially connected obstacles is feasible. Two obstacles p_a and p_b are defined as connected $C(p_a, p_b)$ by

$$C(p_a, p_b) \leftrightarrow \{(r_{x,p_a} - r_{x,p_b})^2 + (r_{y,p_a} - r_{y,p_b})^2 \leq (R_{p_a} + R_{p_b})^2\} \bigvee \{\exists p_c \colon C(p_a, p_c) \wedge C(p_b, p_c)\} \tag{11}$$

Thus, an obstacle p_a is connected to the obstacle p_b if a continuous path exists between any point in the two obstacles which always remains inside an obstacle. These connections between obstacles can be calculated during the initialization of the branch-and-bound method then compared to the diverted path to determine if it is logically feasible. Note that Eq. (11) is a recursive definition and can be applied to chains of overlapping obstacles of any length. If two obstacles p_a and p_b are connected, then the only logically feasible paths with obstacles p_a and p_b active are those where the path is specified to go the same direction around both obstacles, that is, counterclockwise around both p_a and p_b or clockwise around both p_a and p_b:

$$C(p_a, p_b) \rightarrow [\text{direction}(p_a) = \text{direction}(p_b)] \tag{12}$$

A path which is not logically feasible can be immediately fathomed without further solving. For example, there is no point searching for paths clockwise around p_1 and counterclockwise around p_2 if p_1 and p_2 overlap.

If the path is logically feasible, then the search continues to check for dynamic feasibility. To perform the search, the dynamics constraints 2 and 3 are replaced by the following relaxed forms involving new slack decision variables $\gamma^+\gamma^-\delta^+\delta^-$:

$$\delta^-(t) - a_{\max} \leq \frac{\mathrm{d}}{\mathrm{d}t} v_x(t) \leq a_{\max} + \delta^+(t) \tag{13a}$$

$$\delta^-(t) - a_{\max} \le \frac{\mathrm{d}}{\mathrm{d}t} v_y(t) \le a_{\max} + \delta^+(t) \tag{13b}$$

$$\gamma^-(t) + v_{\min}^2 \le v_x(t)^2 + v_y(t)^2 \le v_{\max}^2 + \gamma^+(t) \tag{13c}$$

where

$$\gamma^-(t), \gamma^+(t), \delta^-(t), \delta^+(t) \ge 0 \tag{14}$$

The original objective [Eq. (8)] is replaced so as to seek the minimum γ and δ solution with respect to the shortest-time path goal, where, in effect, the original dynamics model is least violated:

$$J_d^* = \min \int_{t_0}^{t_f} \left(\frac{[\gamma^+(t) + \gamma^-(t)]}{\eta} + \frac{[\delta^+(t) + \delta^-(t)]}{\eta\epsilon} \right) \mathrm{d}t + t_f \tag{15}$$

$$J_g = \int_{t_0}^{t_f} \gamma^+(t) + \gamma^-(t) + \delta^+(t) + \delta^-(t)\, \mathrm{d}t \tag{16}$$

Where ϵ and η are scaling factors, η is gradually reduced toward zero during optimization. The problem is judged to have a dynamically feasible solution (henceforth denoted as the "gamma path") in step 5 if J_g is within some very small tolerance of zero. If no feasible solution exists, the active branch is known to be fathomed.

B. Step 7: Search for Shortest Path

Following the search for a dynamically feasible path in Sec. III.A, the search for the shortest-time path can take place by removing γ and δ from the decision variables again and solving the minimum time problem for the original model in Sec. II. The problem is to minimize the time [Eq.(8)] subject to initial, terminal, bounding, and dynamics constraints [Eqs.(1–5) and (7)] and avoidance of only the active obstacle set A, leading the following modified form of Eq. (6):

$$[r_x(t) - r_{x,p}]^2 + [r_y(t) - r_{y,p}]^2 \ge R_p^2 \quad \forall\ k, \quad \forall\ p \in A \tag{17}$$

C. Steps 10 and 11: Construct Diverted Path

The two paths P_a and P_c are constructed from a combination of the previous path P_s and a newly created diversion clockwise or counterclockwise around obstacle p_b. The time t_c in path P_s is defined as the last time before the path intersects obstacle p_b, and time t_e is defined as the first time after the path has ceased intersecting the obstacle. A path P_a (or P_c) is constructed between $(r_x(t_c), r_y(t_c))$ and $(r_x(t_e), r_y(t_e))$ using an arc with the same radius as obstacle p_b covering the angle θ_c or θ_a between points $(r_x(t_c), r_y(t_c))$ and $(r_x(t_e), r_y(t_e))$ either counterclockwise or clockwise, respectively.

D. Stage-by-Stage Illustration

This section provides a graphical illustration of how the algorithm works. Figure 1 shows the stages involved in solving an example problem using the new algorithm.

Stage 1: Solve the relaxed problem ignoring all obstacles, representing the root node of the search tree.

Stage 2a: "Branch on" obstacle 1 as the first obstacle encountered. Begin with solutions heading counterclockwise around obstacle 1. Generate a diverted path passing counterclockwise around obstacle 1. This path is not a feasible solution, because it involves two sharp corners.

Stage 2b: Find a feasible path counterclockwise around obstacle 1, ignoring all other obstacles, using the optimization described in Sec. III.A (henceforth, any active obstacle is in a darker shade).

Stage 2c: Find the shortest path counterclockwise around obstacle 1, ignoring all other obstacles. Because this path does not intersect with any other obstacles, it is feasible for the complete problem and therefore becomes the incumbent solution. This branch is fathomed, and the algorithm now returns to a previously unexplored branch.

Stage 3: Branch clockwise around obstacle 1. (For brevity, the initial guess step and dynamically feasible solution step are not shown individually.) The solution path continues to intersect obstacles and the algorithm continues branching.

Stage 4: Branch clockwise around obstacle 1 and counterclockwise around obstacle 2. The resulting path is a feasible solution with a length of 10.64, shorter than the incumbent. The incumbent is updated, the branch is fathomed and the algorithm returns to a previously unexplored branch.

Stage 5: Branch clockwise around both obstacle 1 and obstacle 2. The solution path is not feasible overall as it intersects obstacle 3. However, its length is greater than that of the incumbent, and so the branch is fathomed. (Because the path will only get longer as more obstacles are considered, there is no benefit to continuing the search of this branch.)

Final stage: The active subproblem list is empty. The problem is completely solved with a best path length of 10.64.

IV. Branching Strategies

This section describes the nine branching strategies developed and their geometric reasoning. These strategies do not exhaustively cover all possible branching strategies, however, they highlight those suited to the specifics of path planning. In the Algorithm, the two key decision points which define the branching strategy are steps 4 and 9. Step 9 selects which, out of a set of inactive obstacles that the proposed path intersects, will be added to the active set for branching on (or avoiding). The order in which obstacles are added to the active set is very important: those added early into the solution will have a greater effect on the speed of the search for the globally optimal solution [37]. Step 4 will select from the active subproblem list which subproblem to solve next. At step 4, depth-first would call for the selection of the subproblem with the most number of obstacles in its active obstacle set; meanwhile, best-bound would call for the subproblem with the shortest path to the goal. Strategy acronyms are derived using the form "decision at step 9: decision at step 4." Figure 2 shows a graphical representation of eight of the nine strategies developed where the numbers on the obstacles reflect the order in which they are added to the active obstacle list (step 4), and the solutions of the first three subproblems solved are also shown. Tables 1 and 2 summarize the investigated decisions at steps 4 and 9. Note that, although each strategy has some rationale behind it, described in the following subsections, it is in practice impossible to predict which effect will dominate solution time, and so the final choice of strategy is determined by experiment.

Table 1 Summary of which obstacle to add to the active set at step 9

Strategy	Summary
First, F	First inactive obstacle encountered on a path is added.
Least, L	Obstacle with the smallest incursion is added.
Most, M	Obstacle with the largest minimum incursion is added.

Table 2 Summary of which subproblem to next solve at step 4

Strategy	Summary
Random, RND	Subproblem is randomly chosen from the active subproblem list.
One side, ONE	Subproblem with the most number of active obstacles is chosen. In the case of a tie, the problem which heads to the right of the latest obstacle is chosen.
Alternating side, ALT	Subproblem with the most number of active obstacles is chosen. In the case of a tie, the problem which heads to the opposite side of the latest obstacle is chosen.
Best bound, BST	Subproblem with the shortest P_s is chosen. In the event of a tie, an arbitrary side is chosen.
Least diverted, LST	Subproblem which will require the least diversion is chosen.
Most diverted, MST	Subproblem which will require the most diversion is chosen.

A. Strategy F:RND

The F:RND strategy specifies that, in step 9, the first (F) inactive obstacle intersected along a path is chosen as p_b and added to the active obstacle set in the resulting subproblem, and that, in step 4, the next subproblem to be solved is chosen randomly (RND) from the active subproblem list. The rationale for choosing the first obstacle intersected along a path is that this obstacle is very likely to be an active constraint in the final path, whereas it is unlikely that branching on any other obstacle will divert the path out of this obstacle. Observations suggest it is better to branch on such

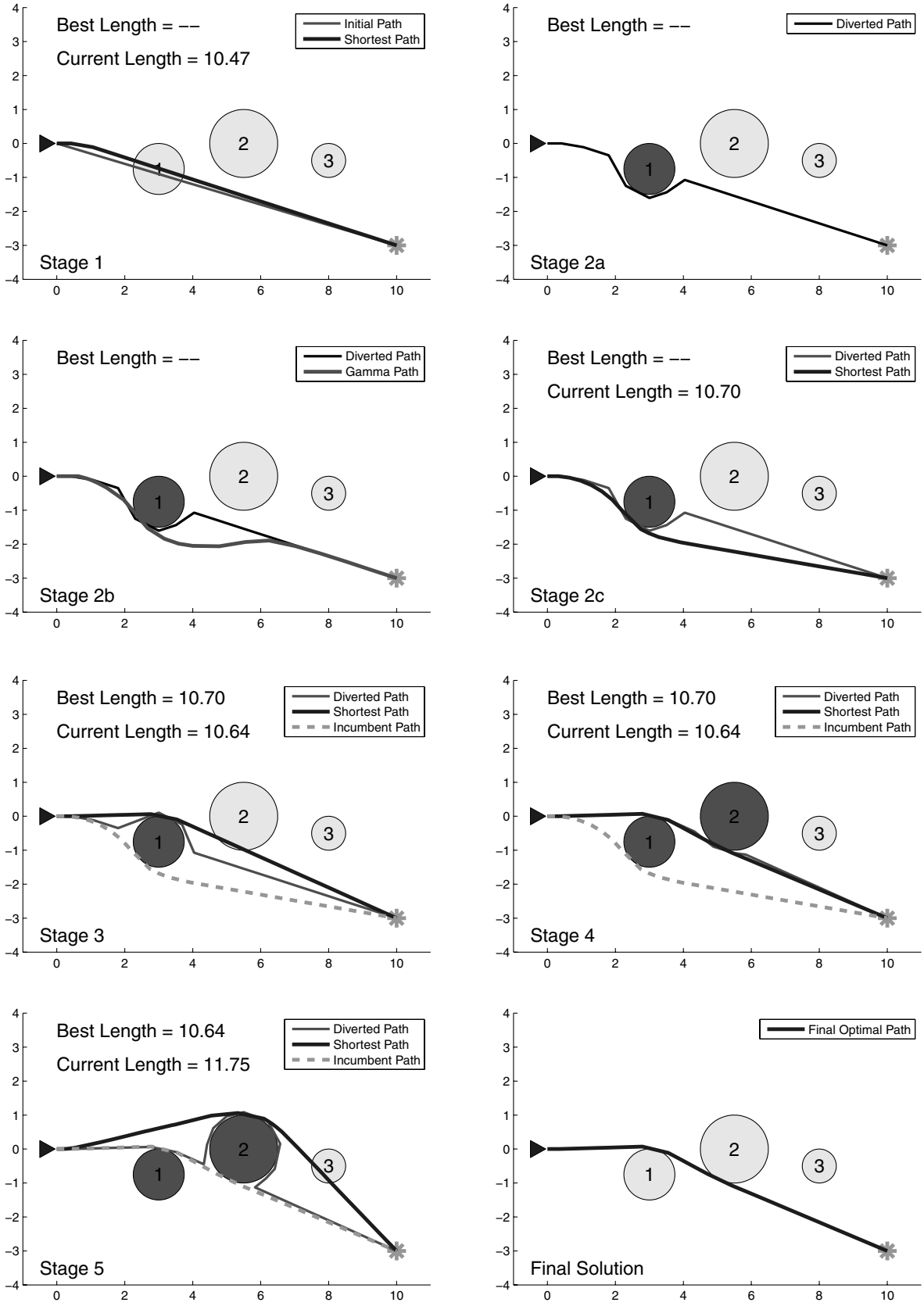


Fig. 1 Illustration of the branch-and-bound concept.

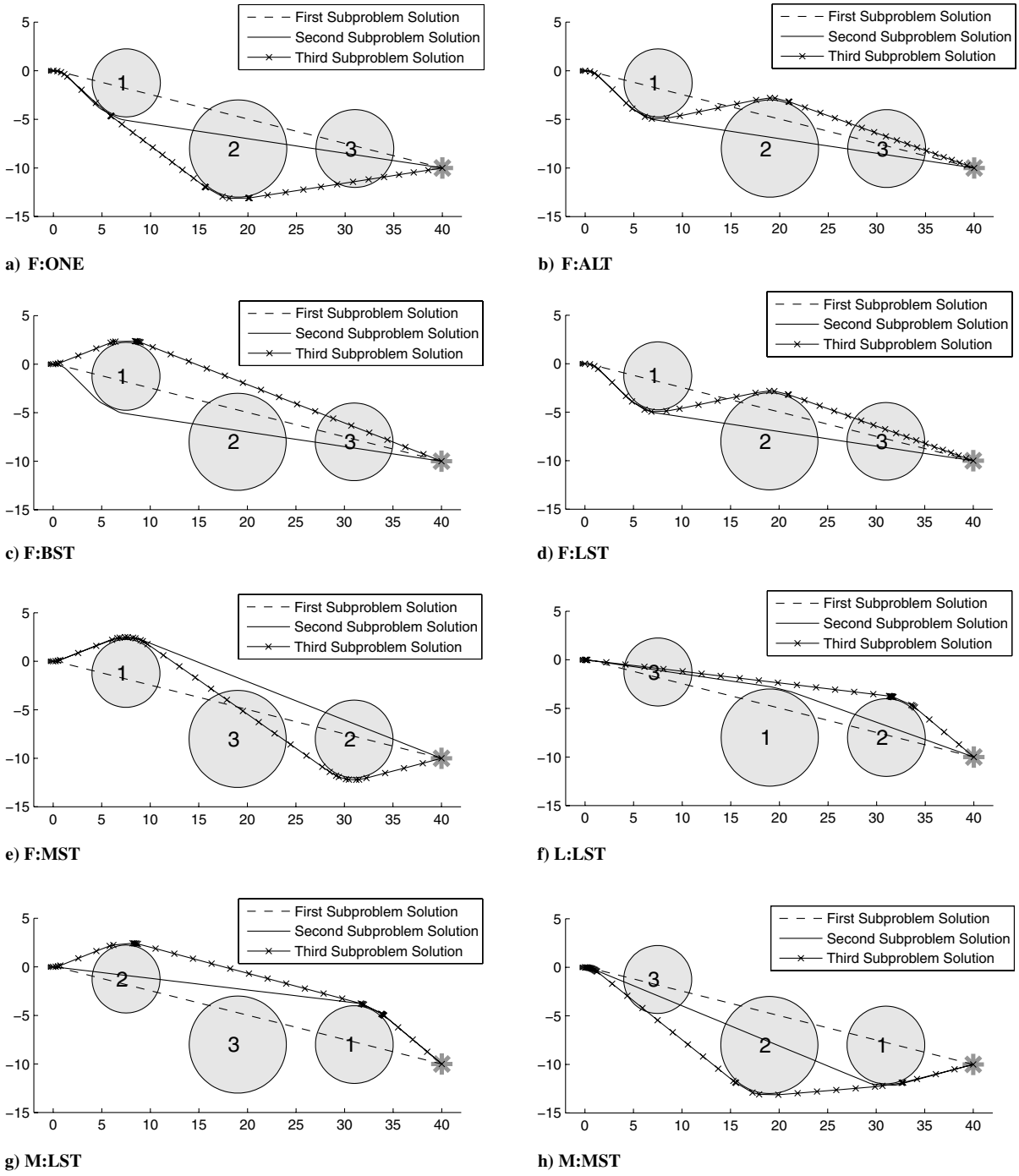


Fig. 2 Illustration of the branching strategies.

"important" obstacles earlier rather than later, because they significantly affect the objective value and can lead to substantial pruning. The F:RND strategy acts as the benchmark for comparisons with all other strategies selecting the first inactive obstacle intersecting a path.

B. Strategy F:ONE

The F:ONE strategy specifies that, in step 4, the next subproblem to be solved from the active subproblem list is chosen as the subproblem with the most active obstacles. The decision in step 4 classifies the strategy as a depth-first strategy. In the case of a tie, the one which is diverted to a particular side of an obstacle ("ONE" solving always to one side first) is chosen. In the implementation for this paper, the chosen side was counterclockwise. This was the strategy employed in the first work on the obstacle branch-and-bound method [36]. The F:ONE strategy seeks to rapidly find a feasible path, so that it can use this to fathom other paths, and does so by diverting around encountered obstacles counterclockwise. In the worst case, this will form a long path which has managed to end up passing counterclockwise around every obstacle in the problem.

C. Strategy F:ALT

The F:ALT strategy aims to improve upon the idea of the F:ONE strategy by looking for a good direct path rather than following wide diversions. Like F:ONE, in step 4, the next subproblem to be solved from the active subproblem list is chosen as the subproblem with the most active obstacles, making this another depth-first strategy. However, in the case of a tie, F:ALT selects the subproblem which will be diverted to a different side than the previously solved subproblem ("ALT" solving to alternating sides). Note that a tie is

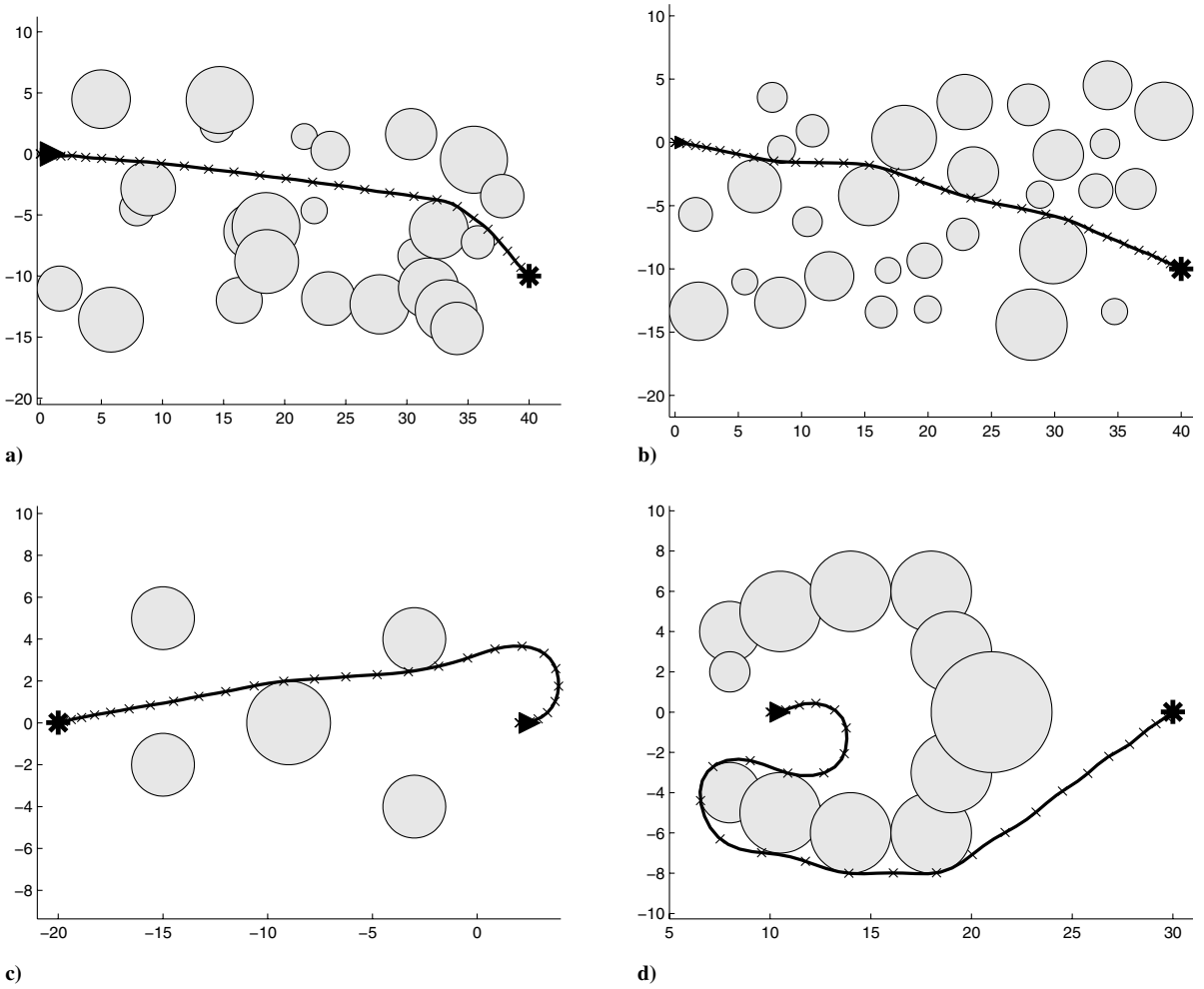


Fig. 3 Solved examples showing the final optimal path.

quite likely, because there is typically a small integer number of obstacles, so that F:ALT and F:ONE differ significantly in practice. For example, if the subproblem selected had the path diverted counterclockwise, the next subproblem selected would be the one that diverted clockwise. Geometrically, this would appear as searching for a path that "wiggles" through the middle of an obstacle field instead of allowing itself to be diverted all the way around on one side.

D. Strategy F:BST

Instead of following a depth-first approach, the F:BST strategy follows a best-bound approach ("BST" solving to previous best-bounded solution). This strategy specifies that, in step 4, the next subproblem to be solved from the active subproblem list is chosen as that with the shortest path to the goal as a parent. In other words, F:BST chooses the subproblem with the shortest P_s from which it was derived in steps 8 and 9. Because the search tree is binary, in the case of a tie, one side is arbitrarily chosen. This follows the logic that, by selecting subproblems with the shortest parent path, the strategy will be focusing on subproblems which look more likely to generate the desired objective function (the shortest path to the goal).

E. Strategy F:LST

For the F:LST strategy, in step 4, the next subproblem to be solved from the active subproblem list is chosen as the subproblem where the path before diversion (i.e., the P_s from which the P_c or P_a was derived) incurred into the newly active obstacle the least ("LST" solving the least diverted subproblem). The incursion of a path P_s into an obstacle, with respect to passing the obstacle on the left or right (thus the incursion associated with P_l or P_r, respectively), is the maximum distance between any point in P_s which is inside the obstacle, perpendicularly to the left or right edge of that obstacle. The geometric argument is that, if a path only slightly clipped one edge of an obstacle, it could mean only a minor adjustment is required to make the entire path feasible and the strategy focuses on making minor adjustments first.

F. Strategy F:MST

The alternative to the F:LST strategy, instead of choosing the subproblem with the least incursion, F:MST chooses the problem with the greatest incursion ("MST" solving the most diverted subproblem). For the F:MST strategy in step 9, the first inactive obstacle intersected by a path will be added to the active obstacle set in the resulting subproblem, and that, in step 4, the next subproblem

Table 3 Table showing the mean and standard (STD) deviation of the number of subproblems solved over 3000 obstacle sets with the overall ranking of the strategies

Strategy	F:LST	L:LST	F:BST	M:LST	F:ALT	F:RND	F:ONE	F:MST	M:MST
Mean	14.12	16.38	16.94	19.28	19.83	22.20	27.28	30.69	35.85
STD	10.76	13.68	10.17	17.71	13.65	14.45	17.24	18.54	23.42
Rank	1	2	3	4	5	6	7	8	9

to be solved from the active subproblem list is chosen as the subproblem where the path before diversion incurred into the newly active obstacle the most. This strategy seeks to solve paths requiring major diversions before those requiring minor diversions.

G. Strategy L:LST

In keeping with the naming convention, L:LST uses the same selection method in step 4 as F:LST, based on the least diverted path. However, in step 9, the obstacle p_b chosen for branching is the inactive obstacle with the smallest incursion (least diversion) from path P_s. Geometrically, this strategy follows a similar argument to F:LST, however, it focuses on making minor adjustments anywhere along a path rather than from start to finish. The disadvantage with this method is that, if an inactive obstacle is intersected almost directly through its center, it and its resulting subproblems will not be added to the active obstacle set until the rest of the smaller incursions have been dealt with.

H. Strategy M:LST

To combat the aforementioned issue, the M:LST problem aims to choose the obstacle p_b in step 9 that has most impact on the path, requiring most diversion ("M" selects the obstacle with the greatest incursion). Specifically, it evaluates the least incursion for each obstacle, and then chooses the obstacle with the largest of those values. Geometrically, this is equivalent to the inactive obstacle whose center is closest to the path P_s. The strategy specifies that, for step 4, it then chooses the least diverted subproblem. This strategy focuses on dealing with obstacles which will have a large effect on the path regardless of where they are along the path.

I. Strategy M:MST

For completeness, the strategy M:MST specifies that, at step 9, for all inactive obstacles currently intersected by a path, their incursion on the side of minimum incursion is calculated and the strategy picks the largest of these incursions to add to the active obstacle set generating two subproblems. This is followed by step 4 where it then chooses the subproblem with the greatest incursion in the active subproblem list.

V. Results

The results in this section were generated by an implementation of the Algorithm, using MATLAB[‡] for the Algorithm implementation, and SNOPT[§] as the nonlinear optimizer through an AMPL[¶] interface on a 2.4 GHz PC with 2 GB RAM.

A. Large-Scale Feature Examples

Figure 3 shows the optimal paths for a pair of large examples and a pair of examples with active nonlinear dynamics constraints. In each figure, the vehicle starts at the triangle with a maximum velocity in the positive x direction; the goal in all figures is represented as a star. Figure 3a demonstrates the method's capability to cope with overlapping obstacles as well as the spaced obstacles shown in Fig. 3b. Figure 3c shows Eq. (3) to be active when turning around for a goal behind the original position. Similarly, Fig. 3d also shows the capability to turn around, in this case to escape what is otherwise a dead end.

B. Comparison of Branching Strategies

Initially, we investigated the computational complexity of the nine branching strategies compared with each other. All strategies were applied to the same set of 3000 randomly generated problems, 300 for each number of obstacles between 15 and 25. In every case, the vehicle's initial and target conditions were the same:

[‡]Information available at http://www.mathworks.com/.

[§]Information available at http://www.sbsi-sol-optimize.com/as/so_product_snopt.htm.

[¶]Information available at http://www.ampl.com/.

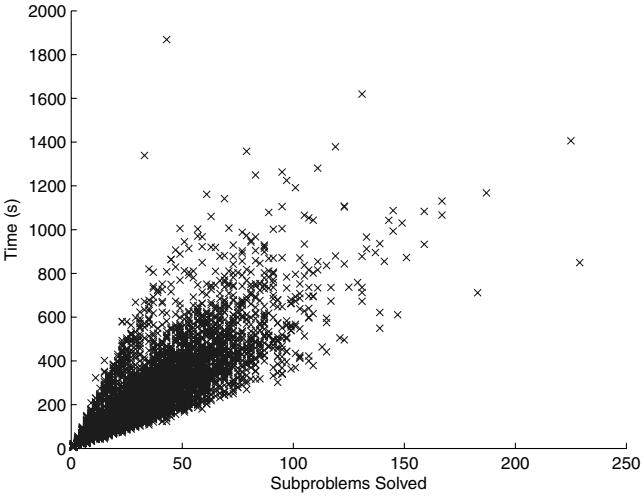


Fig. 4 Correlation of number of subproblems solved and solve time taken.

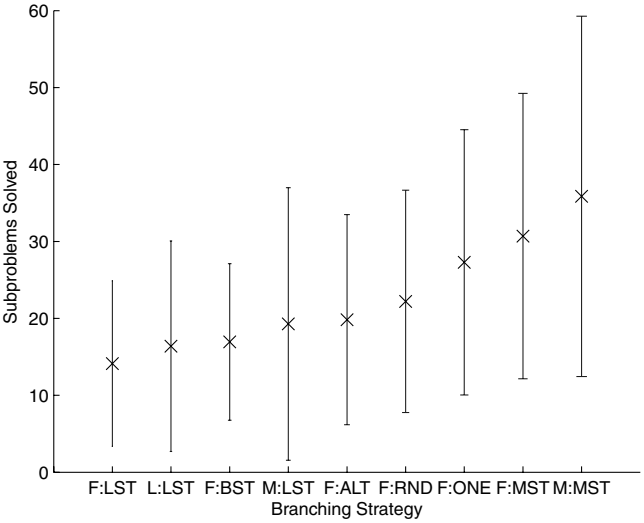


Fig. 5 Comparison of nine branching strategies.

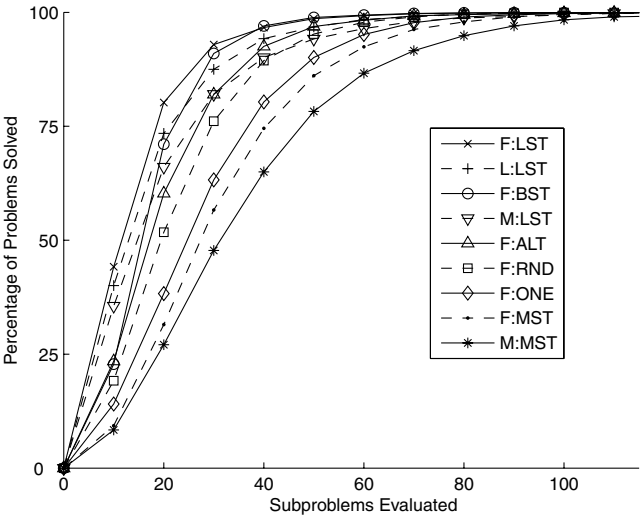
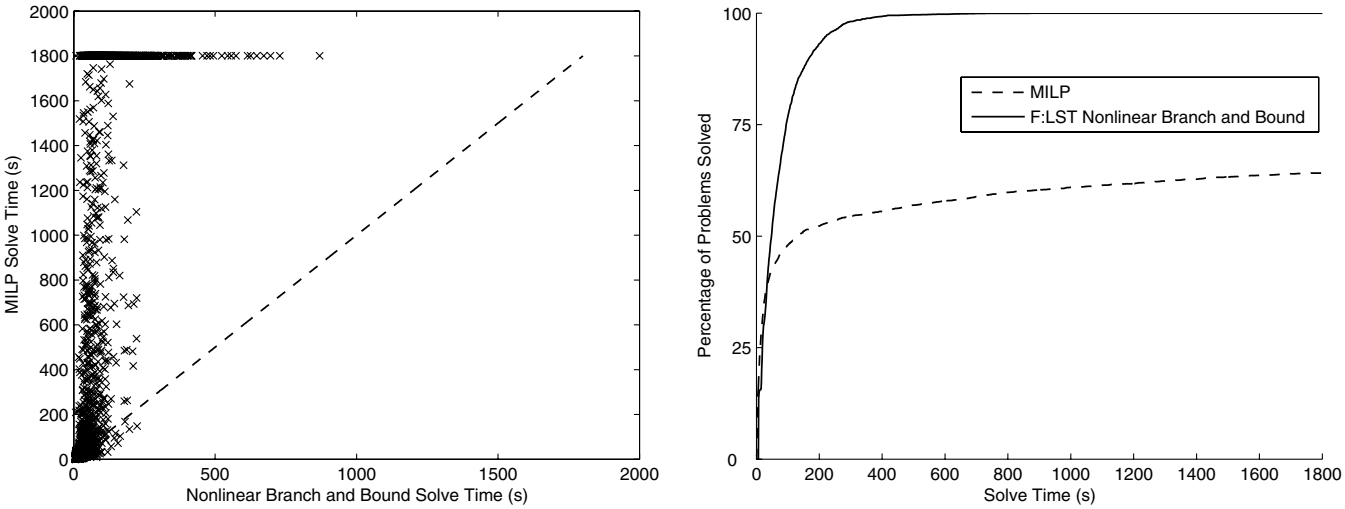
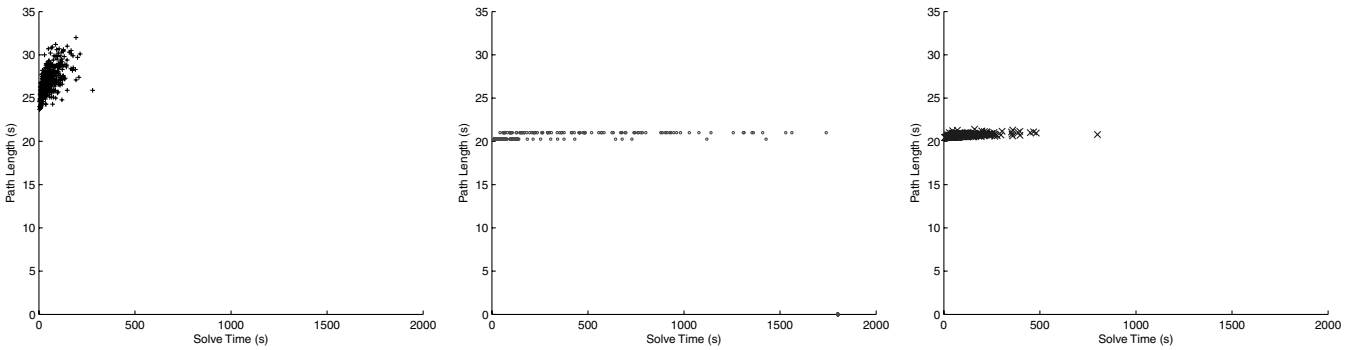


Fig. 6 Comparison of nine branching strategies, number of problems solved against number of subproblems required to solve.



a) b)

Fig. 7 Comparison of F:LST strategy branch-and-bound with MILP.



a) RRT b) MILP c) Nonlinear branch and bound

Fig. 8 Comparison of F:LST strategy branch-and-bound with MILP.

$(r_{x0}, r_{y0}, v_{x0}, v_{y0}) = (0, 0, 3, 0)$ and $(r_{xT}, r_{yT}) = (40, -10)$. The obstacles had random radii between one and two units and their centers were randomly generated within a region joining the start and end points:

$$0 \leq r_{x,p}^{\text{obs}} \leq 40 \qquad -15 \leq r_{y,p}^{\text{obs}} \leq 5 \tag{19}$$

These bounds encourage collisions between the vehicle and the obstacles rather than leaving overly sparse problems to solve. Layouts with obstacles too close to the start or end points were rejected. Out of the 300 problems of each number of obstacles, 150 were cases allowing obstacles to overlap, and the other 150 were cases where overlaps had been prohibited. No time limit was imposed on the solving and no optimization was terminated early, hence the globally optimal solution was found in every case. To identify a hardware independent metric, both the number of subproblems evaluated in the search tree and the time taken for each problem to completely solve were measured. The number of subproblems solved in a search tree directly relates to the speed at which each of the branching strategies converge to the globally optimal solution. Figure 4 demonstrates the trend between the number of subproblems solved and the complete solution time of the problem for 15,000 examples. It is clear from the figure that the solution time and the number of subproblems are highly correlated. However, by using the number of subproblems as the metric, the comparisons are purely based on the branching strategy and are independent of hardware and underlying optimization software.

Table 3 shows the results from the nine branching strategies. The overall ranking values obtained were calculated using a paired T test between the nine strategies. Figure 5 shows the data from Table 3 pictorially with the mean of each strategy marked by an x and the standard deviations plotted as error bars. Figure 6 displays the cumulative percentage of cases completed by each approach against the number of subproblems evaluated to solve the cases. All strategies significantly outperformed the worst case of branch-and-bound tree enumeration where $2^{M+1} - 1$ subproblems would be solved and the entire search tree enumerated (for problems containing 15–25 obstacles, the total subproblems possible lies between approximately 65,000 subproblems and over 67 million subproblems). The primary conclusion from all analyses is that the F:LST strategy is the best, outperforming others both on average and in terms of proportion of problems solved in any given time. Furthermore, the "LST" method for the subproblem selection in step 4 appears to be the most important of the two factors: LST methods came first, second, and fourth out of nine, whereas MST methods came in eighth and ninth. The two MST strategies and the ONE strategy were worse than the random benchmark, F:RND. On average, the best strategy, F:LST, was 30% better than F:RND and 53% better than the slowest strategy, M:MST. The best strategy, F:LST, remains a geometrically rational result. The strategy effectively constructs paths from start to goal, obstacle by obstacle, and chooses the next path to update by the smallest required change.

C. Comparison to Existing Methods

Figure 7a plots the solve time using the nonlinear branch-and-bound approach against the solve time for the same case using the MILP method [20]. A data set of 3000 nonoverlapping obstacle layouts were used. The existing MILP approach results were generated using a discrete time linear dynamics model, which was

then solved using CPLEX** with an AMPL interface on a PC with the same specification as that used for the nonlinear branch-and-bound method. For convenience, a time limit of 1800 s (30 min) was placed on the optimizations. For MILP, the circular obstacles were approximated as octagons. The dashed line marks the point where the two methods' solve times would be equal. The faster cases lie between the origin and 100 s, accounting for roughly 50% of all the cases. Cases in this range were observed to be those in which either minimal obstacles were encountered or the vehicle had a clear path to the target point, and there is minimal difference between the solve times for either approach. The MILP approach had a smaller general computational overhead and thus had the better minimum time of the two methods. Beyond 100 s, the majority of all cases are above the dashed line indicating the branch-and-bound method is solving faster than the MILP. The MILP method timed out on 35% of all problem cases, shown by the clumped line at the top of Fig. 7a. In total, 64% of the generated problems were completed faster by the nonlinear branch-and-bound method and thus lie above the dashed line. Of those completed faster by the MILP method, the majority lie in the initial 100 s cluster and were influenced far more by the overheads associated with each method: the MILP method having an overhead of under a second, and the branch-and-bound method close to 6 s. Harder problems, such as those involving a lot of obstacles in the way of the goal, were solved comfortably by the branch-and-bound method, whereas MILP struggled to complete within the time limit. Both methods produce the same paths to within a small tolerance, allowing for the linearization of the circular obstacles and the differing dynamics models between the two methods.

To explore the results further, Fig. 7b shows the cumulative percentage of cases completed by each approach against time in seconds. The F:LST nonlinear branch-and-bound method managed to complete 100% of all cases compared to 64.3% of cases solved by MILP before the time limit. The MILP method quickly plateaus after 1 min, whereas the nonlinear branch-and-bound method continues rapidly toward its maximum.

Figure 8 shows a comparison of path length and solve times associated with 420 individual problems solved by three independent methods: nonlinear branch and bound, MILP, and rapidly exploring random trees (RRTs) [14]. The problem cases were 20 cases of 10–30 nonoverlapping obstacles. The RRTs method results were obtained from the average of five runs of the RRTs on each problem case. Path lengths for MILP and branch-and-bound methods were very close, which was to be expected because both find the global optimal for each case. The discrete time measure of the MILP model reflects in Fig. 8b from the two distinct values for path length. A time limit of 1800 s was imposed on the MILP optimization and, if a problem case exceeded this time limit, then the path length was recorded as zero.

RRTs demonstrated the fastest solve times but found suboptimal paths. Allowing the RRTs to search longer for better paths could have improved this but increased the solve time. The branch-and-bound method had a higher time overhead than RRTs and was somewhat slower overall, however branch-and-bound achieved a significant improvement on path length. Branch-and-bound had a drastic improvement on solve time for MILP and acted competitively with the RRT solve time. Thus, it can be seen that the branch-and-bound method offers a combination of the fast solution times of RRTs and the high performance of MILP.

VI. Conclusions

In this paper, we have demonstrated that branch-and-bound optimization can be directly applied to the problem of obstacle avoidance for a vehicle with a nonlinear dynamics model. The method we have developed can calculate globally optimal solutions subject to reasonable assumptions. We have further demonstrated that the choice of branching strategy is of key importance in the use of branch-and-bound optimization. The first encountered, least diverted (F:LST) branching strategy was found to be the fastest out of the nine strategies investigated, and geometric reasoning behind the result has been presented. We have further demonstrated through simulation of two banks of problem cases with 1–30 obstacles that the F:LST branch-and-bound method can be significantly faster on solve time compared to an existing mixed-integer linear programming approach and, in a three-way comparison, offers significant benefits to path length while remaining competitive on solve time with rapidly exploring random trees. In particular, the computational advantage for the branch-and-bound method was greater in the most difficult problems.

**Information available at http://www.ilog.com/products/cplex/.

Acknowledgments

The authors would like to thank Geoffrey T. Huntington for his help with the understanding and implementation of the Gauss pseudospectral method.

References

[1] Pachter, M., and Chandler, P. R., "Challenges of Autonomous Control," *IEEE Control Systems Magazine*, Vol. 18, No. 4, 1998, pp. 92–97. doi:10.1109/37.710883

[2] Perry, T. S., "In Search of the Future of Air Traffic Control," *IEEE Spectrum*, Vol. 34, No. 8, 1997, p. 19. doi:10.1109/6.609472

[3] Reddien, G. W., "Collocation at Gauss Points as a Discretization in Optimal Control," *SIAM Journal on Control and Optimization*, Vol. 17, No. 2, 1979, pp. 298–306. doi:10.1137/0317023

[4] Latombe, J.-C., *Robot Motion Planning*, Kluwer Academic, Dordrecht, The Netherlands, 1991, pp. 33–37.

[5] Raghunathan, A. U., Gopal, V., Subramanian, D., Biegler, L. T., and Samad, T., "Dynamic Optimization Strategies for 3D Conflict Resolution of Multiple Aircraft," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 4, 2004, pp. 586–594. doi:10.2514/1.11168

[6] Singh, G., and Hadaegh, F. Y., "Collision Avoidance Guidance for Formation Flying Applications," *Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit*, AIAA Paper 2001-4088, 2001.

[7] Roger, A. B., and McInnes, C. R., "Safety Constrained Free-Flyer Path Planning at the International Space Station," *Journal of Guidance, Control, and Dynamics*, Vol. 23, No. 6, Dec. 2000, pp. 971–979. doi:10.2514/2.4656

[8] Ghosh, R., and Tomlin, C., "Maneuver Design for Multiple Aircraft Conflict Resolution," *Proceedings of the American Control Conference*, American Automatic Control Council, Evanston, IL, June 2000, pp. 672–676.

[9] Bortoff, S. A., "Path-Planning for UAVs," *Proceedings of the 2000 American Control Conference*, Vol. 1, No. 6, June 2000, pp. 364–368. doi:10.1109/ACC.2000.878915

[10] McLain, T. W., and Beard, R. W., "Trajectory Planning for Coordinated Rendezvous of Unmanned Air Vehicles," *Proceedings of the AIAA Guidance, Navigation and Control Conference*, AIAA Paper 2000-4369, 2000.

[11] Bicchi, A., and Pallottino, L., "On Optimal Cooperative Conflict Resolution for Air Traffic Management Systems," *IEEE Transactions On Intelligent Transportation Systems: A Publication Of The IEEE Intelligent Transportation Systems Council*, Vol. 1, No. 4, 2000, pp. 221–232. doi:10.1109/6979.898228

[12] Milam, M. B., Mushambi, K., and Murray, R. M., "A New Computational Approach to Real-Time Trajectory Generation for Constrained Mechanical Systems," *Proceedings of the IEEE Conference on Decision and Control*, Vol. 1, Inst. of Electrical and Electronics Engineers, New York, 2000, pp. 845–851. doi:10.1109/CDC.2000.912875

[13] Barraquand, J., Kavraki, L., Latombe, J.-C., Motwani, R., and Raghavan, P., "A Random Sampling Scheme for Path Planning," *International Journal of Robotics Research*, Vol. 16, No. 6, 1997, pp. 759–774. doi:10.1177/027836499701600604

[14] Lee, Y. I., and Kouvaritakis, B., "Constrained Receding Horizon Predictive Control for Systems with Disturbances," *International Journal of Control*, Vol. 72, No. 11, 1999, pp. 1027–1032. doi:10.1080/002071799220579

[15] Frazzoli, E., Dahleh, M., and Feron, E., "Real-Time Motion Planning for Agile Autonomous Vehicles," *Proceedings of the AIAA Guidance,*

Navigation and Control Conference, AIAA Paper 2000-4056, Aug. 2000.

[16] Chandler, P. R., Pachter, M., and Rasmussen, S., "UAV Cooperative Control," *Proceedings of American Control Conference*, Vol. 1, American Automatic Control Council, Evanston, IL, June 2001, pp. 50–55. doi:10.1109/ACC.2001.945512

[17] Bellingham, J., Richards, A., and How, J. P., "Receding Horizon Control of Autonomous Aerial Vehicles," *Proceedings of the American Control Conference*, Vol. 5, American Automatic Control Council, Evanston, IL, May 2002, pp. 3741–3746. doi:10.1109/ACC.2002.1024509

[18] Richards, A., Schouwenaars, T., How, J., and Feron, E., "Spacecraft Trajectory Planning with Avoidance Constraints Using Mixed-Integer Linear Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 4, 2002, pp. 755–764. doi:10.2514/2.4943

[19] Borrelli, F., Subramanian, D., Raghunathan, A. U., and Biegler, L. T., "MILP and NLP Techniques for Centralized Trajectory Planning of Multiple Unmanned Air Vehicles," *Proceedings of the 2006 American Control Conference*, American Automatic Control Council, Evanston, IL, June 2006, pp. 5763–5768. doi:10.1109/ACC.2006.1657644

[20] Richards, A., and How, J., "Aircraft Trajectory Planning with Collision Avoidance Using Mixed Integer Linear Programming," *Proceedings of the American Control Conference*, Vol. 3, American Automatic Control Council, Evanston, IL, May 2002, pp. 1936–1941. doi:10.1109/ACC.2002.1023918

[21] Bertsimas, D., and Tsitsiklis, J. N., *Introduction to Linear Optimization*, Athena Scientific, Belmont, MA, 1997, pp. 485–490.

[22] Dakin, R. J., "A Tree-Search Algorithm for Mixed Integer Programming Problems," *Computer Journal*, Vol. 8, No. 3, 1965, pp. 250–255. doi:10.1093/comjnl/8.3.250

[23] Geoffrion, A. M., "Integer Programming by Implicit Enumeration and Balas Method," *SIAM Review*, Vol. 9, No. 2, 1967, pp. 178–190. doi:10.1137/1009031

[24] Golomb, S. W., and Baumert, L. D., "Backtrack Programming," *Journal of the Association for Computing Machinery*, Vol. 12, No. 4, 1965, pp. 516–524. doi:10.1145/321296.321300

[25] Glover, F., "A Multiphase-Dual Algorithm for the Zero-One Integer Programming Problem," *Operations Research*, Vol. 13, No. 6, 1965, pp. 879–919. doi:10.1287/opre.13.6.879

[26] Balas, E., "Discrete Programming by the Filter Method," *Operations Research*, Vol. 15, No. 5, 1967, pp. 915–957. doi:10.1287/opre.15.5.915

[27] Lawler, E. L., and Wood, D. E., "Branch-And-Bound Methods: A Survey," *Operations Research*, Vol. 14, No. 4, 1966, pp. 699–719. doi:10.1287/opre.14.4.699

[28] Nilsson, N. J., *Problem-Solving Methods in Artificial Intelligence*, McGraw, New York, 1971, pp. 43–77.

[29] Ibaraki, T., "Theoretical Comparisons of Search Strategies in Branch-And-Bound Algorithms," *International Journal of Parallel Programming*, Vol. 5, No. 4, 1976, pp. 315–344. doi:10.1007/BF00998631

[30] Benichou, M., Gauthier, J. M., Girodet, P., Hentges, G., Ribiere, G., and Vincent, O., "Experiments in Mixed-Integer Linear Programming," *Mathematical Programming*, Vol. 1, No. 1, 1971, pp. 76–94. doi:10.1007/BF01584074

[31] Gupta, O. K., and Ravindan, A., "Branch and Bound Experiments in Convex Nonlinear Integer Programming," *Management Science*, Vol. 31, No. 12, Dec. 1985, pp. 1533–1546. doi:10.1287/mnsc.31.12.1533

[32] Kindt, V. T., Croce, F. D., and Esswein, C., "Revisiting Branch and Bound Search Strategies for Machine Scheduling Problems," *Journal of Scheduling*, Vol. 7, No. 6, 2004, pp. 429–440. doi:10.1023/B:JOSH.0000046075.73409.7d

[33] Hartley, R., *Linear and Nonlinear Programming*, Ellis Horwood, Chichester, England, U.K., 1985, pp. 167–175.

[34] Huntington, G., "Advancement and Analysis of a Guass Pseudospectral Transcription for Optimal Control Problems," Ph.D. Thesis, Dept. of Aeronautics and Astronautics, Massachusetts Inst. of Technology, Cambridge, MA, May 2007.

[35] Ross, M., and Fahroo, F., "Direct Trajectory Optimization by a Chebyshev Pseudospectral Method," *Proceedings of the American Control Conference*, Vol. 6, American Automatic Control Council, Evanston, IL, June 2000, pp. 3860–3864. doi:10.1109/ACC.2000.876945

[36] Eele, A., and Richards, A., "Path-Planning with Avoidance Using Nonlinear Branch-and-Bound Optimisation," *Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit*, AIAA Paper 2007-6793, Aug. 2007.

[37] Glover, F., and Tangedahl, L., "Dynamic Strategies for Branch and Bound," *OMEGA: International Journal of Management Science*, Vol. 4, No. 5, 1976, pp. 571–576. doi:10.1016/0305-0483(76)90007-4