An Accelerator for Packages Solving Discrete-Time Optimal Control Problems ⋆

Hoam Chung, Elijah Polak, and Shankar Sastry *

* *Authors are with the Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA, 94720-1770 USA e-mail: hachung/polak/sastry at eecs.berkeley.edu*

Abstract: We present an accelerator scheme for use with existing packages that solve nonlinear programming problems with a large number of inequality constraints that arise in the process of discretizing continuous-time optimal control problems with state-space constraints. This scheme is based on the concept of outer approximations used in semi-infinite programming and acts as an *external*, active constraints set strategy.
Our scheme constructs a finite sequence of inequality constrained nonlinear programming problems, containing a progressively larger subset of the constraints in the original problem, and submits these problems to a nonlinear programming solver for a fixed number of iterations. We prove that this scheme computes a solution of the original problem and show, by means of numerical experiments, that it results in reductions in computing time ranging from a factor of 6 to a factor of over 400.

Keywords: Optimal control, nonlinear programming, active-set strategies.

1. INTRODUCTION

Optimal control problems with state space constraints are usually solved by discretizing the dynamics, which results in the conversion of the continuous-time optimal control problem into a discrete-time optimal control problem with many inequality constraints. A discrete-time optimal control problem is a nonlinear programming problem and hence can be solved by nonlinear programming algorithms. The distinguishing features of these nonlinear programming problems are (i) the required gradients can be computed using adjoint equations, and (ii) although they have very large numbers of inequality constraints, relatively few of these inequalities are active.

An important example of optimal control problems with state space constraints arises in the control of unmanned aerial vehicles (UAV's) using receding horizon control (RHC). RHC is a form of sample-data control that determines the control to be applied over the next sampling interval by solving an optimal control problem during the current sampling interval. The optimal control problems for RHC control of UAV's are characterized by large numbers of collision avoidance inequalities and by expensive evaluations of the gradients of these inequality defining functions. Since potential collisions are confined to relatively short segments of the UAV trajectories, most of the collision avoidance inequalities are inactive. In the case of UAV's, the sampling intervals are short and hence the viability of the RHC scheme is largely determined by the speed of the nonlinear programming solvers.

Unfortunately, most standard nonlinear programming packages, including the excellent set found in TOMLAB Holmström et al. [2006], including SNOPT by Murray et al. [2002], NPSOL by Gill et al. [1998], Schittkowski SQP by Schittkowski [1982], and KNITRO [1] by Byrd et al. [2006], are not designed to exploit the fact that a problem with a large number of nonlinear inequality constraints may have few active constraints.

In this paper, we present an accelerator scheme for general nonlinear programming packages, based on the theory of outer approximations (see Polak [1997], p. 460), which acts as an *external* active constraint set strategy for solving nonlinear programming problems with a large number of inequality constraints. This strategy constructs a finite sequence of inequality constrained nonlinear programming problems, containing a progressively larger subset of the constraints in the original problem, and submits them to a nonlinear programming solver for a fixed number of iterations. We prove that this scheme computes a solution of the original problem and show, by means of numerical experiments, that when applied to UAV control, this strategy results in reductions in computing time ranging from a factor of 6 to a factor of over 400. Our new strategy is particularly effective when used with nonlinear programming solvers that allow warm starts. It may be useful to observe that a related strategy in Polak et al. [2007] for solving semi-infinite minimax problems using log-sum-exponential smoothing has proved to be equally effective.

In Section 2 we present a motivational optimal control example, in Section 3 we state our strategy in the form of

⋆ This work is supported by ARO SWARMS (W911NF-0510219), ONR MURI (N00014-02-1- 0720), and ARO Phase II STTR (W911NF-06-C-0192)

[1] KNITRO is a collection of optimization algorithms, and we use the algorithm option 'Interior/Direct' with quasi-Newton symmetric rank one updates in this paper.

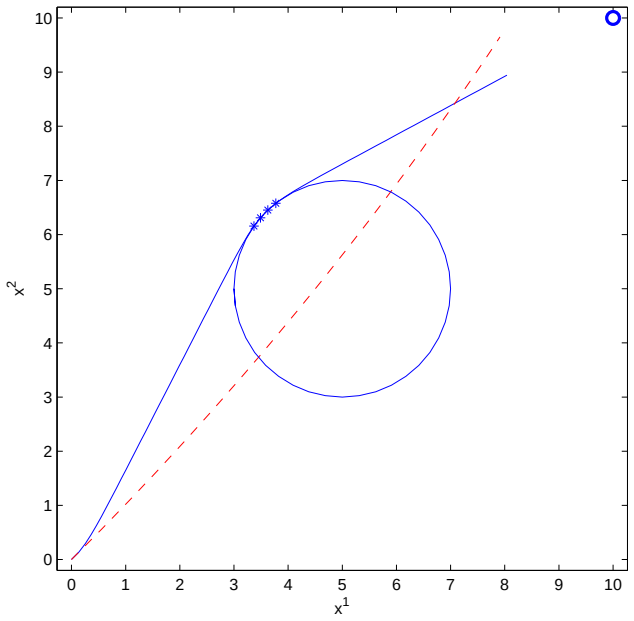

Fig. 1. Initial trajectory (dashed red) and optimal trajectory (solid blue). Active constraints (constraints within feasibility tolerance) are marked as '*'.

an algorithm and provide theoretical justification for it, in Section 4 we present numerical results, and our concluding remarks are in Section 5.

Notation *We will denote elements of a vector by superscripts (e.g., x^i) and elements of a sequence or a set by subscripts (e.g., η_k).* □

2. OPTIMAL CONTROL EXAMPLE

In order to motivate our approach, consider the following simple, fixed-time optimal control problem, which consists of minimizing the sum of the energy used by a UAV and the square of the distance to a desired destination point. The UAV trajectory is required to stay out of a circle. As we will see, while the number of state space constrains is as large as the number of points used to discretize the dynamics, only those that touch the forbidden circle are active. A geometric representation of the constraints is shown in Fig. 1, which presents the given initial trajectory and the computed optimal trajectory for the example below.

For the sake of simplicity, we assume that the UAV flies at a constant speed v and that the scalar control u determines the yaw rate of the UAV. In order to state the optimal control problem as an end-point problem defined on $[0,1]$, we rescale the state dynamics using the actual terminal time T and augment the 3-dimensional physical state with a fourth component, x^4, so that

$$x^4(t) = \int_0^t \frac{T}{2} u^2(\tau) d\tau \tag{1}$$

represents the energy used. The resulting dynamics have the form

$$\frac{dx}{dt} = \begin{bmatrix} Tv\cos x^3 \\ Tv\sin x^3 \\ Tu \\ \frac{T}{2}u^2 \end{bmatrix} \triangleq h(x(t),u(t)) \tag{2}$$

with the initial state $x(0)$ given. We will denote the solution of the dynamic equation (2) by $x(t,u)$, with $t \in [0,1]$. The optimal control problem now assumes the form

$$\min_{u\in\mathbb{R}} f^0(u) \triangleq x^4(1,u) + (x^1(1,u)-10)^2 + (x^2(1,u)-10)^2, \tag{3}$$

subject to:

$$f^t(u) \triangleq (x^1(t,u)-5)^2 + (x^2(t,u)-5)^2 \geq 2^2,\ t\in[0,1]. \tag{4}$$

In order to solve this problem, we must discretize the dynamics. We use Euler's method to obtain

$$\bar{x}(t_{k+1}) - \bar{x}(t_k) = \Delta h(\bar{x}(t_k),\bar{u}(t_k)),\ \bar{x}(0) = x(0), \tag{5}$$

with $\Delta \triangleq 1/N$, $N\in\mathbb{N}$, $t_k \triangleq k\Delta$ and $k\in\{0,1,\ldots,N\}$. We use an overbar to distinguish between the exact variables and the discretized variables. We will denote the solution of the discretized dynamics by $\bar{x}(t_k,\bar{u})$, $k=0,1,\ldots,N$. Letting

$$\bar{u} \triangleq (\bar{u}(t_0),\bar{u}(t_1),\ldots,\bar{u}(t_{N-1})), \tag{6}$$

we obtain the discretized optimal control problem

$$\min_{\bar{u}\in\mathbb{R}^N} \bar{f}^0(\bar{u}) \triangleq \bar{x}^4(1,\bar{u}) + (\bar{x}^1(1,\bar{u})-10)^2 + (\bar{x}^2(1,\bar{u})-10)^2, \tag{7}$$

subject to the constraints

$$\bar{f}^k(\bar{u}) \triangleq -(\bar{x}^1(t_k,\bar{u})-5)^2 - (\bar{x}^2(t_k,\bar{u})-5)^2 + 2^2 \leq 0,\quad k\in\{1,\ldots,N\}. \tag{8}$$

Clearly, (7), (8) is a mathematical programming problem which is distinguished from ordinary mathematical programming problems only by the fact that adjoint equations can be used in the computation of the gradients of the functions $f^k(\cdot)$, $k=0,1,\ldots,N$.

3. THE ALGORITHM

We now proceed in a more abstract setting. Consider the inequality constrained minimization problem:

$$\mathbf{P_q} \qquad \min\{f^0(\eta) \mid f^j(\eta)\leq 0, j\in\mathbf{q}\}, \tag{9}$$

where $\eta\in\mathbb{R}^n$, and $\mathbf{q}\triangleq\{1,\ldots,q\}$. We assume that functions $f^j:\mathbb{R}^n\to\mathbb{R}$ are at least once continuously differentiable. In the context of discrete optimal control problems, η can be either a discretized control sequence or an initial state - discretized control sequence pair [2].

Next we define the index set $\mathbf{q}_\epsilon(\eta)$ with $\epsilon>0$ by

$$\mathbf{q}_\epsilon(\eta) \triangleq \{j\in\mathbf{q} \mid f^j(\eta) \geq \psi_+(\eta) - \epsilon\}, \tag{10}$$

where

$$\psi(\eta) \triangleq \max_{j\in\mathbf{q}} f^j(\eta), \tag{11}$$

[2] see Polak [1997] Chapter 4 for a detailed treatment of discrete time optimal control problems

and

$$\psi_+(\eta) \stackrel{\triangle}{=} \max\{0, \psi(\eta)\}. \quad (12)$$

Definition 1. We say that an algorithm defined by a recursion of the form

$$\eta_{k+1} = A(\eta_k), \quad (13)$$

for solving inequality constrained problems of the form (9), is *convergent* if any accumulation point of a sequence $\{\eta_i\}_{i=0}^{\infty}$, constructed according to the recursion (13), is a feasible stationary point. □

Finally, we assume that we have a convergent algorithm for solving inequality constrained problems of the form (9), represented by the recursion function $A(\cdot)$, i.e., given a point η_k the algorithm constructs its successor η_{k+1} according to the rule (13).

Algorithm 2: Active-Set Algorithm for Inequality Constrained Minimization Problems.

Data: η_0, $\epsilon > 0$, $N_{iter} \in \mathbb{N}$
Step 0: Set $i = 0$, $\mathbf{Q}_0 = \mathbf{q}_\epsilon(\eta_0)$.
Step 1: Set $\zeta_0 = \eta_i$ and perform N_{iter} iterations of the form $\zeta_{k+1} = A(\zeta_k)$ on the problem

$$\mathbf{P}_{\mathbf{Q}_i} \qquad \min\{f^0(\zeta) | f^j(\zeta) \le 0,\ j \in \mathbf{Q}_i\} \quad (14)$$

to obtain $\zeta_{N_{iter}}$ and set $\eta_{i+1} = \zeta_{N_{iter}}$.
Step 2: Compute $\psi(\eta_{i+1})$.
if $\zeta_{N_{iter}}$ is returned as a global, local, or stationary solution of $\mathbf{P}_{\mathbf{Q}_i}$ and $\psi(\eta_{i+1}) \le 0$, **then**
  STOP,
else
  Compute

$$\mathbf{Q}_{i+1} = \mathbf{Q}_i \cup \mathbf{q}_\epsilon(\eta_{i+1}), \quad (15)$$

  and set $i = i + 1$, and go to Step 1.
end if

Lemma 3. Suppose that $\epsilon > 0$ and that the sequence $\{\eta_i\}_{i=0}^{\infty}$, in $\mathbb{R}^n$, is such that $\eta_i \to \hat{\eta}$ as $i \to \infty$. Then there exists an i_0 such that for all $i \ge i_0$, $\mathbf{q}_0(\hat{\eta}) \subset \mathbf{q}_\epsilon(\hat{\eta})$.

Proof. By definition (10), for any $j \in \mathbf{q}_0(\hat{\eta})$,

$$f^j(\hat{\eta}) - \psi_+(\hat{\eta}) \le 0. \quad (16)$$

First suppose that $\psi_+(\hat{\eta}) = \psi(\hat{\eta}) \ge 0$. Then the set $\mathbf{q}_0(\hat{\eta})$ is nonempty. Since all the functions $f^j(\cdot)$ and $\psi(\cdot)$ are continuous, $[f^j(\eta_i) - \psi_+(\eta_i)] \to [f^j(\hat{\eta}) - \psi_+(\hat{\eta})]$ as $i \to \infty$. Hence there must exist an i_0 such that for all $i \ge i_0$ and $j \in \mathbf{q}_0(\hat{\eta})$,

$$f^j(\hat{\eta}) - \psi_+(\hat{\eta}) \ge -\epsilon, \quad (17)$$

which proves that for all $i \ge i_0$, $\mathbf{q}_0(\hat{\eta}) \subset \mathbf{q}_\epsilon(\hat{\eta})$.

Next suppose that $\psi(\hat{\eta}) < 0$. Then $\psi_+(\hat{\eta}) = 0$ and the set $\mathbf{q}_0(\hat{\eta})$ is empty. Since the empty set is a subset of any set, the desired result follows.

Lemma 4. Suppose that $\epsilon > 0$ and that the sequence $\{\eta_i\}_{i=0}^{\infty}$, in $\mathbb{R}^n$, is such that $\eta_i \to \hat{\eta}$ as $i \to \infty$ and that $\mathbf{Q} = \cup_{i=0}^{\infty} \mathbf{q}_\epsilon(\eta_i) \subset \mathbf{q}$. For any $\eta \in \mathbb{R}^n$, let

$$\psi_{\mathbf{Q}}(\hat{\eta}) = \max_{j \in \mathbf{Q}} f^j(\eta). \quad (18)$$

If $\psi_{\mathbf{Q}}(\hat{\eta}) \le 0$, then $\psi(\hat{\eta}) \le 0$.

Proof. By Lemma 3, $\mathbf{q}_0(\hat{\eta}) \subset \mathbf{Q}$. Since by assumption, $f^j(\hat{\eta}) \le 0$ for all $j \in \mathbf{Q}$, it follows that $\psi(\hat{\eta}) \le 0$.

Lemma 5. Suppose that $\mathbf{Q} \subset \mathbf{q}$ and consider the problem $\mathbf{P}_{\mathbf{Q}}$

$$\min\{f^0(\eta) | f^j(\eta) \le 0, j \in \mathbf{Q}\}. \quad (19)$$

Suppose that $\hat{\eta} \in \mathbb{R}^n$ is feasible for $\mathbf{P}_{\mathbf{q}}$, i.e, $f^j(\eta) \le 0$ for all $j \in \mathbf{q}$.

(a) If $\hat{\eta}$ is a global minimizer for $\mathbf{P}_{\mathbf{Q}}$, then it is also a global minimizer for $\mathbf{P}_{\mathbf{q}}$.
(b) If $\hat{\eta}$ is a local minimizer for $\mathbf{P}_{\mathbf{Q}}$, then it is also a local minimizer for $\mathbf{P}_{\mathbf{q}}$.
(c) If $\hat{\eta}$ is a stationary point for $\mathbf{P}_{\mathbf{Q}}$, i.e., it satisfies the F. John conditions in John [1948] (or Theorem 2.2.4, p. 188 in Polak [1997]), then it is also a stationary point for $\mathbf{P}_{\mathbf{q}}$.

Proof. Clearly, since $\hat{\eta}$ is feasible for $\mathbf{P}_{\mathbf{q}}$ it is also feasible for $\mathbf{P}_{\mathbf{Q}}$.

(a) Suppose that $\hat{\eta}$ is not a global minimizer for $\mathbf{P}_{\mathbf{q}}$. Then there exists an η^* such that $f^j(\eta^*) \le 0$ for all $j \in \mathbf{q}$ and $f^0(\eta^*) < f^0(\hat{\eta})$. Now, η^* is also feasible for $\mathbf{P}_{\mathbf{Q}}$ and hence $\hat{\eta}$ cannot be a global minimizer for $\mathbf{P}_{\mathbf{Q}}$, a contradiction.

(b) Suppose that $\hat{\eta}$ is not a local minimizer for $\mathbf{P}_{\mathbf{q}}$. Then there exists a sequence $\{\eta_i\}_{i=0}^{\infty}$ such that $\eta_i \to \hat{\eta}$, $f^0(\eta_i) < f^0(\hat{\eta})$ and $f^j(\eta_i) \le 0$ for all i and $j \in \mathbf{q}$. But this contradicts the assumption that $\hat{\eta}$ is a local minimizer for $\mathbf{P}_{\mathbf{Q}}$.

(c) Since $\hat{\eta}$ satisfies the F. John conditions for $\mathbf{P}_{\mathbf{Q}}$, there exist multipliers $\mu^0 \ge 0$, $\mu^j \ge 0$, $j \in \mathbf{Q}$, such that $\mu^0 + \sum_{j \in Q} \mu^j = 1$,

$$\mu^0 \nabla f^0(\hat{\eta}) + \sum_{j \in Q} \mu^j \nabla f^j(\hat{\eta}) = 0 \quad (20)$$

and

$$\sum_{j \in Q} \mu^j f^j(\hat{\eta}) = 0. \quad (21)$$

Clearly, $\hat{\eta}$ also satisfies the F. John conditions for $\mathbf{P}_{\mathbf{q}}$ with multipliers $\mu^j = 0$ for all $j \notin \mathbf{Q}$ and otherwise as for $\mathbf{P}_{\mathbf{Q}}$.

Combining the above lemmas, we get the following convergence result.

Theorem 6. Suppose that the problem $\mathbf{P}_{\mathbf{q}}$ has feasible solutions, i.e., there exist vectors η^* such that $f^j(\eta^*) \le 0$ for all $j \in \mathbf{q}$.

(a) If Algorithm 2 constructs a finite sequence $\{\eta_i\}_{i=0}^{k}$, exiting in Step 2, with $i + 1 = k$, then η_k is a global, local, or stationary solution for $\mathbf{P}_{\mathbf{q}}$, depending on the exit message from the solver defined by $A(\cdot)$.
(b) If $\{\eta_i\}_{i=0}^{\infty}$ is an infinite sequence constructed by Algorithm 2 in solving $\mathbf{P}_{\mathbf{q}}$. Then any accumulation point [3] $\hat{\eta}$ of this sequence is feasible and stationary for $\mathbf{P}_{\mathbf{q}}$.

Proof. (a) If sequence $\{\eta_i\}_{i=0}^{k}$ is finite, then, by the exit rule, it is feasible for $\mathbf{P}_{\mathbf{q}}$ and it is a global, local, or stationary solution for $\mathbf{P}_{\mathbf{Q}_i}$. It now follows from Lemma 4, that it is also a global, local, or stationary solution for $\mathbf{P}_{\mathbf{q}}$.
(b) Since the sets $\mathbf{Q}_i$ grow monotonically, and since $\mathbf{q}$ is finite, there must exist an i_0 and a set $\mathbf{Q} \subset \mathbf{q}$, such

[3] A point $\hat{\eta}$ is said to be an accumulation point of the sequence $\{\eta_i\}_{i=0}^{\infty}$, if there exists an infinite subsequence, indexed by $K \subset \mathbb{N}$, $\{\eta_i\}_{i \in K}$, such that $\eta_i \stackrel{K}{\longrightarrow} \hat{\eta}$. as $i \stackrel{K}{\longrightarrow} \infty$

that $\mathbf{Q}_i = \mathbf{Q}$ for all $i \geq i_0$. Next, it follows from the fact that $A(\cdot)$ is convergent, that for any accumulation point $\hat{\eta}$, $\psi_{\mathbf{Q}}(\hat{\eta}) \leq 0$ and hence, from Lemma 4 that $\psi(\hat{\eta}) \leq 0$, i.e., that $\hat{\eta}$ is a feasible point for $\mathbf{P_q}$. Since for any accumulation point $\hat{\eta}$, by Lemma 3, $\mathbf{q}_0(\hat{\eta}) \subset \mathbf{Q}$, it now follows from the fact that $A(\cdot)$ is convergent and Lemma 5 that any accumulation point $\hat{\eta}$ is stationary for $\mathbf{P_q}$.

4. NUMERICAL RESULTS

All numerical experiments were performed using MATLAB V7.2 and TOMLAB V5.5 Holmström et al. [2006] in Windows XP, on a desktop computer with an Intel Xeon 3.2GHz processor with 3GB RAM. Optimization solvers tested in this paper were the Schittkowski SQP algorithm with cubic line search Schittkowski [1982], NPSOL 5.02 Gill et al. [1998], SNOPT 6.2 Murray et al. [2002], and KNITRO Byrd et al. [2006]. It should be clear from the form of Algorithm 2, that our strategy benefits considerably from warm starts of the nonlinear programming solvers to be used after constructing the active set $\mathbf{Q}_i$. Hence it is desirable to use solvers with as extensive a warm start capability as possible, so that one can transmit the last value of important information from the last iteration of a solver on the problem $\mathbf{P}_{\mathbf{Q}_i}$ as initial conditions for solving the problem $\mathbf{P}_{\mathbf{Q}_{i+1}}$. SNOPT allows the user to provide initial variables and their states and slack variables. NPSOL allows the user to provide initial variables and their states, Lagrange multipliers, as well as an initial Hessian approximation matrix for quasi-Newton updates. conSolve, the TOMLAB implementation of the Schittkowski SQP algorithm, allows the user to provide an initial solution vector and initial Hessian matrix approximation. KNITRO allows the user to provide only the initial solution vector. For maximum efficiency, this data must be saved at the end of the $i-$th run and transmitted as initial data for the $i+1$-th run of the solver.

As we will see from our numerical results, the performance of Algorithm 2 is much more sensitive to the parameters N_{iter} and ϵ when using a solver, such as KNITRO, that has minimal warm start features, than when it is using a solver with extensive warm start features.

4.1 Single UAV Example

Our first set of results are for the optimal control problem (7), (8) described in Section 2, with initial state $x_0 = (0, 0, \pi/4, 0)$. All computations were initialized using the control $\bar{u}_0 = 8 \times 10^{-3}\mathbf{1}_{1\times N}$. The Parameters used in the numerical experiments were $T = 25$, $v = 0.5$, and $N = 64$, thus there were 64 inequality constraints (8) in the discretized optimal control problem.

The numerical results are summarized in the Tables 1 - 4. In these tables, N_{grad}, the total number of gradient evaluations, and t_{CPU}, the total CPU time for achieving an optimal solution using Algorithm 2, are defined as follows:

Table 1. External Active-Set Strategy with Schittkowski SQP Algorithm, Single UAV Example.

ϵ	N_{iter}	i_T	f^0	N_{grad}	$\lvert\mathbf{Q}\rvert$	t_{CPU}	$\%_{Raw}$
1	10	13	5.0367	7996	27	46.4	109.3
1	20	5	5.0367	5953	21	32.8	77.3
1	30	3	5.0367	3112	16	17.8	41.8
0.1	10	10	5.0367	1478	8	11.2	26.5
0.1	20	6	5.0367	1757	8	12.9	30.4
0.1	30	3	5.0367	986	8	7.59	17.9
0.01	10	12	5.0367	1114	5	10.4	24.4
0.01	20	5	5.0367	752	6	7.06	16.6
0.01	30	4	5.0367	676	5	7.55	17.8
Raw			5.0367	9600	64	42.5	100

Table 2. External Active-Set Strategy with NPSOL, Single UAV Example.

ϵ	N_{iter}	i_T	f^0	N_{grad}	$\lvert\mathbf{Q}\rvert$	t_{CPU}	$\%_{Raw}$
1	10	11	5.0367	3214	16	18.2	38.0
1	20	5	5.0367	2625	16	14.2	29.7
1	30	3	5.0367	2113	16	11.3	23.5
0.1	10	12	5.0367	1578	8	10.5	22.0
0.1	20	6	5.0367	1411	8	9.05	18.9
0.1	30	4	5.0367	1465	8	9.17	19.1
0.01	10	12	5.0367	915	5	7.30	15.2
0.01	20	7	5.0367	920	5	7.32	15.3
0.01	30	5	5.0367	808	5	6.73	14.0
Raw			5.0367	11136	64	47.9	100

Table 3. External Active-Set Strategy with SNOPT, Single UAV Example.

ϵ	N_{iter}	i_T	f^0	N_{grad}	$\lvert\mathbf{Q}\rvert$	t_{CPU}	$\%_{Raw}$
1	10	9	5.0367	2196	16	12.7	50.5
1	20	6	5.0367	1871	16	10.5	41.7
1	30	2	5.0367	893	16	4.87	19.4
0.1	10	7	5.0367	732	8	4.96	19.8
0.1	20	6	5.0367	808	8	5.43	21.6
0.1	30	2	5.0364	794	7	5.16	20.6
0.01	10	9	5.0367	575	5	4.64	18.5
0.01	20	6	5.0367	395	5	3.43	13.7
0.01	30	5	5.0367	500	5	4.29	17.1
Raw			5.0367	5760	64	25.1	100

$$N_{grad} = \sum_{i=0}^{i_T} |Q_i| \times \left\{ \begin{matrix} \text{number of gradient function calls} \\ \text{during } i\text{-th inner iteration} \end{matrix} \right\}$$

$$t_{CPU} = \sum_{i=0}^{i_T} \Big[\left\{ \begin{matrix} \text{CPU time spent for} \\ i\text{-th inner iteration} \end{matrix} \right\} + \left\{ \begin{matrix} \text{CPU time spent for} \\ \text{setting up } i\text{-th inner iteration} \end{matrix} \right\} \Big]. \tag{22}$$

In the above, and in the tables, i_T is the value of the iteration index i at which Algorithm 2 is terminated by the termination tests incorporated in the optimization solver used. Also, $\%_{RAW}$, the percentage of t_{CPU} with respect to the computation time with the *raw* algorithm, i.e. using the solver with the full set of constraints (shown in the last row of each table), is used in tables.

Observations.

- For all solvers, there exist parameter pairs of ϵ and N_{iter} which result in more than an 80% reduction in

Table 4. External Active-Set Strategy with KNITRO, Single UAV Example.

ϵ	N_{iter}	i_T	f^0	N_{grad}	$\lvert\mathbf{Q}\rvert$	t_{CPU}	$\%_{Raw}$
1	50	60	5.0369	48689	16	301	1187
1	75	1	5.0367	770	11	5.23	20.6
1	100	1	5.0367	770	11	5.19	20.5
0.1	50	2	5.0367	492	7	4.72	18.6
0.1	75	2	5.0367	492	7	4.72	18.6
0.1	100	2	5.0367	492	7	4.79	18.9
0.01	50	4	5.0367	376	5	5.62	22.1
0.01	75	4	5.0367	376	5	5.57	22.0
0.01	100	4	5.0367	376	5	5.59	22.0
Raw			5.0367	5568	64	25.4	100

total CPU time, as compared to using these solvers without our active-set strategy.

- When using an optimization solver without an extensive warm start functionality, the performance of Algorithm 2 is much more sensitive to the value of N_{iter} than when using a solver with an extensive warm start functionality. Thus with SNOPT and NPSOL, Algorithm 2 is not very sensitive to the values of N_{iter}. With conSolve, Algorithm 2 yields better performance when $N_{iter} \geq 10$ is given. With KNITRO, Algorithm 2 does not work at all well with $N_{iter} \leq 40$, but it was possible to achieve an 80% reduction in computing time (similar to the other solvers), as compared to the direct use of KNITRO, with proper values of N_{iter} and ϵ. (Table 4).

4.2 Multi-UAV Example

The next example consists of controlling multiple UAVs. Suppose that we have N_a UAVs, each with the same dynamics as (2), and we want them to stay in a circular region centered at the origin, without incurring any collisions. The stay-in-a-circle constraints are described by the following equations:

$$f_{\text{bnd}}^{t,i}(u^i) \triangleq x^{1,i}(t,u^i)^2 + x^{2,i}(t,u^i)^2 \leq r_{\text{bnd}}^2,\ t \in [0,1], \quad (23)$$

where $i \in \{1,2,\ldots,N_a\}$ denotes the UAV index. The collision avoidance constraints are given by

$$\begin{aligned} f_{\text{ca}}^{t,(i,j)}(u^i,u^j) \triangleq\ & (x^{1,i}(t,u^i) - x^{1,j}(t,u^j))^2 \\ & + (x^{2,i}(t,u^i) - x^{2,j}(t,u^j))^2 \geq r_{\text{ca}}^2, \\ & t \in [0,1], \end{aligned} \quad (24)$$

where (i,j) is an element of the set of all 2-combinations of the index set $\{1,2,\ldots,N_a\}$. After discretization, as in the previous example, the constraints become

$$\bar{f}_{\text{bnd}}^{k,i}(\bar{u}^i) \triangleq \bar{x}^{1,i}(t,\bar{u}^i)^2 + \bar{x}^{2,i}(t,\bar{u}^i)^2 \leq r_{\text{bnd}}^2,\ k \in \{1,\ldots,N\}, \quad (25)$$

and

$$\begin{aligned} f_{\text{ca}}^{k,(i,j)}(\bar{u}^i,\bar{u}^j) \triangleq\ & (\bar{x}^{1,i}(k,\bar{u}^i) - \bar{x}^{1,j}(k,\bar{u}^j))^2 \\ & + (\bar{x}^{2,i}(k,\bar{u}^i) - \bar{x}^{2,j}(k,\bar{u}^j))^2 \geq r_{\text{ca}}^2, \\ & k \in \{1,\ldots,N\}. \end{aligned} \quad (26)$$

Finally, we obtain the following discretized optimal control problem for the multi-UAV example:

$$\min_{\bar{u}^i \in \mathbb{R}^N,\ i \in \{1,\ldots,N_a\}} \bar{f}^0(\bar{u}) \triangleq \sum_{i=1}^{N_a} \bar{x}^{4,i}(1,\bar{u}^i) \quad (27)$$

Table 5. External Active-Set Strategy with Schittkowski SQP Algorithm, Eight-UAV example. '*' indicates that no meaningful data is available since the algorithm returns without obtaining a solution after $i = 100$ iterations.

ϵ	N_{iter}	i_T	f^0	N_{grad}	$\lvert\mathbf{Q}\rvert$	t_{CPU}	$\%_{Raw}$
1	10	74	10.635	1.1E6	1196	16573	55.1
1	20	22	2.0452	102133	258	1841.3	6.12
1	30	19	3.0388	112905	299	2192.4	7.28
0.1	10	71	3.2144	223197	334	3932.0	13.2
0.1	20	53	3.4327	221146	244	4212.3	14.0
0.1	30	42	2.9046	171308	190	3695.9	12.3
0.01	10	64	2.8008	51016	91	1429.5	4.75
0.01	20	82	4.2251	266843	144	5687.7	18.9
0.01	30	100	*	*	*	*	*
Raw			4.1973	2.6E6	2304	30105	100

subject to the dynamics of each UAV and the constraints (25) and (26). The total number of inequality constraints in this problem is $NN_a(N_a-1)/2 + NN_a$.

The dynamics of the UAV's are nonlinear and the non-collision constraints are non-convex. Hence this problem has many local minima. Consequently, the solution trajectory may depend on the initial control and on the evolution of the active sets during computation.

For numerical experiments, we set $r_{\text{bnd}} = 4$, $r_{\text{ca}} = 1$, $T = 25$, $N = 64$ and $N_a = 8$, resulting in 2304 nonlinear inequality constraints. The initial conditions and initial controls for each agent are set as

$$\begin{aligned} x_0^1 &= (2.5, 2.5, \pi, 0), & \bar{u}_0^1 &= -1.25 \times 10^{-1}\mathbf{1}_{1\times N} \\ x_0^2 &= (-2.5, 2, -\pi/2, 0), & \bar{u}_0^2 &= 1.25 \times 10^{-1}\mathbf{1}_{1\times N} \\ x_0^3 &= (-2.5, -2.5, -\pi/4, 0), & \bar{u}_0^3 &= 1.25 \times 10^{-1}\mathbf{1}_{1\times N} \\ x_0^4 &= (2, -2.5, \pi/2, 0), & \bar{u}_0^4 &= 2.50 \times 10^{-1}\mathbf{1}_{1\times N} \\ x_0^5 &= (2.5, 0, \pi/2, 0), & \bar{u}_0^5 &= 2.50 \times 10^{-1}\mathbf{1}_{1\times N} \\ x_0^6 &= (-2.5, 0, -\pi/2, 0), & \bar{u}_0^6 &= 1.25 \times 10^{-1}\mathbf{1}_{1\times N} \\ x_0^7 &= (0, 3, -3\pi/4, 0), & \bar{u}_0^7 &= 1.25 \times 10^{-1}\mathbf{1}_{1\times N} \\ x_0^8 &= (0, -3, \pi/4, 0), & \bar{u}_0^8 &= -2.50 \times 10^{-1}\mathbf{1}_{1\times N}. \end{aligned} \quad (28)$$

Fig. 2 shows a locally optimal solution for this problem. There are only 16 active constraints at the end. These are all associated with staying in the circle; there are no active non-collision constraints. When properly adjusted, Algorithm 2 accumulates fewer than 16 constraints. Consequently, the reduction in the number of gradient computations is huge, and the savings in CPU time is more dramatic than in the single UAV case. There exist parameter pairs with conSolve and KNITRO that achieve more than 95% savings in computation time. As can be seen from our tables, in several cases using NPSOL and SNOPT, our algorithm used less than 1/400 of the CPU time required by NPSOL or SNOPT to solve the example problem directly, i.e., using the full set of constraints.

5. CONCLUSION

We have presented an external active-set strategy for solving discrete-time optimal control problems with state-space constraints, using nonlinear programming solvers. Our numerical results show that this strategy results in

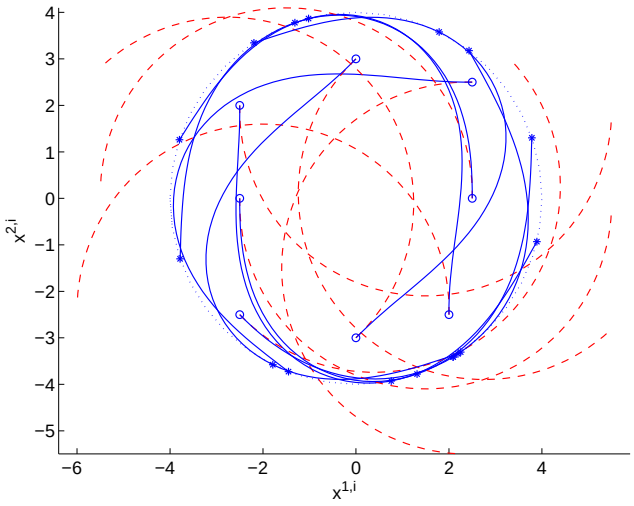

Fig. 2. Initial trajectory (dashed red) and an optimal trajectory (solid blue). Bounding circular region is represented by the dotted blue circle. Active constraints (constraints within feasibility tolerance) are marked as '*' and initial positions are marked as 'o'.

Table 6. External Active-Set Strategy with NPSOL, Eight-UAV example

ϵ	N_{iter}	i_T	f^0	N_{grad}	$\|\mathbf{Q}\|$	t_{CPU}	$\%_{Raw}$
1	10	17	2.1366	28392	214	397.3	0.50
1	20	14	2.1601	46012	229	716.68	0.90
1	30	12	2.1256	66952	239	1072.2	1.34
0.1	10	21	1.7028	11549	64	223.24	0.28
0.1	20	17	1.7028	18001	64	335.97	0.42
0.1	30	14	1.7028	19698	60	374.66	0.47
0.01	10	20	1.7028	5654	31	137.25	0.17
0.01	20	19	1.7028	11888	34	268.34	0.34
0.01	30	19	1.7028	15199	34	339.93	0.43
Raw			2.6809	7.1E6	2304	79817.2	100

Table 7. External Active-Set Strategy with SNOPT, Eight-UAV example

ϵ	N_{iter}	i_T	f^0	N_{grad}	$\|\mathbf{Q}\|$	t_{CPU}	$\%_{Raw}$
1	10	10	2.0874	10840	177	156.81	0.48
1	20	10	2.0897	12503	162	179.88	0.55
1	30	10	2.0840	14923	165	211.98	0.65
0.1	10	20	2.0838	12165	81	205.04	0.63
0.1	20	19	2.0838	13458	81	228.25	0.70
0.1	30	20	2.0838	14471	81	239.79	0.74
0.01	10	18	1.7028	3142	34	70.043	0.22
0.01	20	18	1.7028	3189	34	70.680	0.22
0.01	30	18	1.7028	3189	34	70.744	0.22
Raw			4.1381	3.2E6	2304	32425.7	100

considerable savings in computer time. In our examples, the savings ranged from a factor ranging from 6 to 9, on a problem with 64 constraints, to a factor ranging from around 20 to a factor around of 400 on a problem with 2304 constraints. The results depend on the nonlinear programming solver. There is reason to believe that the larger the number of inequalities in the discrete-time optimal control problem, the larger the computational savings will be. This observation is consistent with the two examples presented in this paper. Finally, it should be obvious that one can add refinements to our algorithm, such as restarting it after a certain number of iterations,

Table 8. External Active-Set Strategy with KNITRO, Eight-UAV Example.

ϵ	N_{iter}	i_T	f^0	N_{grad}	$\|\mathbf{Q}\|$	t_{CPU}	$\%_{Raw}$
1	100	100	*	*	*	*	*
1	200	100	*	*	*	*	*
1	300	100	*	*	*	*	*
0.1	100	100	*	*	*	*	*
0.1	200	23	1.7028	183058	71	3495	7.8
0.1	300	15	1.7028	74569	67	1636	3.7
0.01	100	100	*	*	*	*	*
0.01	200	17	1.7028	32389	34	893.8	2.0
0.01	300	17	1.7028	32389	34	898.8	2.0
Raw			3.7896	4343040	2304	44728	100

so as to avoid accumulating constraints that are not close to being active at the solution.

ACKNOWLEDGEMENTS

The authors wish to thank Prof. Michael Saunders of Stanford for his advice on warm start of NPSOL and SNOPT. We are also grateful to Marcus Edvall at TOMLAB Optimization Inc. for his advice on our codes.

REFERENCES

R. H. Byrd, J. Nocedal, and R. A. Waltz. KNITRO: An integrated package for nonlinear optimization. In G. di Pillo and M. Roma, editors, *Large-Scale Nonlinear Optimization*, pages 35–59. Springer-Verlag, 2006.

Philip E. Gill, Walter Murray, Michael A. Saunders, and Margaret H. Wright. User's guide for NPSOL 5.0: A fortran package for nonlinear programming. Technical Report SOL 86-2, Systems Optimization Laboratory, Department of Operations Research, Stanford University, 1998.

Kenneth Holmström, Anders O. Göran, and Marcus M. Edvall. *User's Guide for TOMLAB*. Tomlab Optimization Inc., December 2006.

F. John. Extremum problems with inequlaities as side condtions. In K. O. Friedrichs, O. W. Neugebauer, and J. J. Stoker, editors, *Studies and Essays: Courant Anniversary Volume*, pages 187–204. Interscience Publishers, Inc., New York, 1948.

W. Murray, P. E. Gill, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12:979–1006, 2002.

E. Polak. *Optimization: Algorithms and Consistent Approximations*, volume 124 of *Applied Mathematical Sciences*. Springer, 1997.

E. Polak, R. S. Womersley, and H. X. Yin. An algorithm based on active sets and smoothing for discretized semi-infinite minimax problems. *Journal of Optimization Theory and Applications*, 2007. in press.

K. Schittkowski. On the convergence of a sequential quadratic programming method with an augmented lagrangian line search function. Technical report, Systems Optimization laboratory, Stanford University, 1982.