Temporal Planning by a Continuous and Differentiable Nonlinear Optimization Formulation and Constraint Partitioning

Yixin Chen
Department of Computer Science and Engineering
Washington University in St Louis
St Louis, MO 63130
E-mail: *chen@cse.wustl.edu*

Abstract

In this paper, we study efficient temporal planning based on a continuous and differentiable nonlinear programming transformation of the planning problem. Based on the observation that many large planning problems have constraint locality, we have previously proposed the constraint partitioning approach that utilizes the constraint structure by partitioning the constraints of a planning problem into subproblems and solving each subproblem individually. Constraint partitioning has led to the design of SGPlan, a state-of-the-art planner. However, SGPlan is based on a mixed-integer programming formulation that is computationally expensive to solve. In this paper, we present a continuous and differentiable nonlinear programming formulation for planning problems, and apply the constraint partitioning approach to this formulation. Because the nonlinear programming transformation is continuous and differentiable, we can utilize powerful existing continuous nonlinear optimization packages to solve each subproblem very quickly. We apply the new strategies in solving some planning benchmark problems and demonstrate significant improvements in time and quality.

1 Introduction

A temporal planning problem involves arranging actions and assigning resources in order to accomplish a given set of tasks over a period of time and to optimize one or more objectives. It can be defined loosely by a set of states; a discrete or continuous time horizon; a set of actions defining valid transitions between states; a set of effects to be evaluated in each state or action; a set of constraints to be satisfied in each state or throughout an action; and a set of goals to be achieved.

Our goal in this paper is to develop efficient search methods for solving large-scale temporal planning problems. In our approach, we formulate a planning problem as a **continuous** (involving continuous variables) and **differentiable** (involving differentiable objective and constraint functions) **nonlinear programming (CNLP)** problem. Based on the subgoals of the planning problem, we partition those constraints related to a subgoal into a disjoint subset and apply the newly developed partitioned search algorithm to solve each subproblem individually before composing the solution.

One key observation we have made is that many planning problems have highly structured constraints derived from the underlying application [26; 4]. For example, in the AIRPORT domain from the Fourth International Planning Competition (IPC4) [9], constraints are largely localized by the subgoals, which means that most constraints only relate one subgoal that aims at moving an airplane around. This is intuitively true because that the movement of an airplane is independent with others most of the time and interacts with others only when the airplane is at the same location as another. Our study on this domain shows that 86 percent of the constraints are local constraints related to a single subgoal [4].

Based on the observation that application-based planning problems have highly localized constraints, we have previously proposed **the constraint partitioning approach** that exploits the constraint structure by partitioning the problem constraints into locally coherent subsets and resolving them individually. The key benefit of the constraint partitioning approach is that it leads to much relaxed subproblems (with fewer constraints) that are significantly easier to solve. However, this approach leads to global constraints relating multiple subproblems that may be violated when the solutions are composed.

We have recently developed the theory of extended saddle points (ESP) for efficiently resolving the global constraints under constraint partitioning. By formulating the constraints into an ℓ-1 penalty formulation, the ESP theory provides a necessary and sufficient condition to prune the search space during the backtracking search

for resolving global constraints.

1.1 SGPlan planner and limitations

We have developed a fully automated PDDL2.2 planner, called SGPlan, based on the constraint partitioning approach [5; 4]. The SGPlan planner formulates a planning problem as a mixed-integer nonlinear programming (MINLP) problem. formulation assumes that some or all of the constraints of a planning problem can be partitioned into $N+1$ stages, each scheduled at different times. Stage t, $t = 0, \ldots, N$, has u_t variables, m_t local equality constraints, and r_t local inequality constraints. Such partitioning decomposes the variable vector $z \in Z$ into $N+1$ subvectors $z(0), \ldots, z(N)$, where $z(t) = (z_1(t), \ldots, z_{u_t}(t))^T$ is a u_t-element *state vector* in mixed space and stage t, and $z_i(t)$ is the i^{th} *dynamic state variable*. The MINLP formulation is as follows:

$$(P_t): \quad \min_z \quad J(z) \qquad (1)$$
$$\text{subject to} \quad h^{(t)}(z(t)) = 0, \quad g^{(t)}(z(t)) \le 0,$$
$$\text{and} \quad H(z) = 0, \quad G(z) \le 0.$$

Here, $h^{(t)} = (h_1^{(t)}, \ldots, h_{m_t}^{(t)})^T$ and $g^{(t)} = (g_1^{(t)}, \ldots, g_{r_t}^{(t)})^T$ are local-constraint functions that involve $z(t)$ in stage t; and $H = (H_1, \ldots, H_p)^T$ and $G = (G_1, \ldots, G_q)^T$ are global-constraint functions that involve state variables in two or more stages. A solution to (1) is a *plan* that consists of the assignment of z.

Based on the ESP theory, the constraint partitioning approach solves (1) by dividing it into smaller subproblems, solving each independently, and resolving those violated global constraints at the end. Though the complexity of resolving the global constraints is exponential and depends on the Cartesian product of the solution spaces of all the subproblems, the base of the exponential complexity is significantly reduced by limiting the solution space of each subproblem using the ESP conditions [26].

The SGPlan planner, using the constraint partitioning approach and powered by the ESP theory, has been successfully applied to solve the PDDL2.2 planning problems in the Fourth International Planning Competition (IPC4), 2004 [9; 4]. SGPlan has won the First Prize in the Suboptimal Temporal Metric Track and the Second Prize in the Suboptimal Propositional Track in IPC4. This achievement has clearly demonstrated the power of the constraint partitioning approach.

Despite its success, the current method used by SGPlan still suffers the following limitations. First, SGPlan uses a mixed-integer MINLP formulation where the functions are not continuous nor differentiable. In general, MINLPs are computationally much more difficult to solve than CNLPs because CNLPs can be solved efficiently using the gradient information of functions whereas MINLPs can not. In fact, SGPlan solves the MINLP subproblems by transforming them back into smaller planning problems and solving them using Metric-FF [15], an existing heuristic planner. Second, SGPlan does not provide any guarantee of the solution quality, since that the subproblems cannot be solved optimally.

In this paper, we solve the above two issues by proposing a continuous and differentiable nonlinear programming (CNLP) formulation of the temporal planning problems. Applying the constraint partitioning approach to this formulation, we derive continuous and differentiable subproblems that can be solved much more efficiently by the state-of-the-art CNLP solvers. Moreover, since the CNLP problems can be solve optimally, the solution quality of the constraint partitioing approach can be significantly improved.

This paper is organized as follows. We first present necessary background of mathematical programming in Section 2. We then present the CNLP formulation of a temporal planning problem in Section 3 and apply the constraint-partitioning approach on the CNLP formulation in Section 4. We present experimental results in Section 5, discuss related work in Section 6, and conclude in Section 7.

2 Mathematical Programming Background

We first overview the basic definitions and methods for CNLP problems. We then overview the recently developed ESP theory that supports the constraint partitioning approach. Refer to [4] for more details.

2.1 Continuous nonlinear optimization

A *continuous nonlinear programming* (CNLP) problem is defined as follows, with continuous and differentiable f, $h = (h_1, \ldots, h_m)^T$, and $g = (g_1, \ldots, g_r)^T$ defined in real space $\mathcal{R}$:

$$(P_c): \quad \min_x \ f(x) \ \text{ where } x = (x_1, \ldots, x_v)^T \in \mathcal{R}^v \qquad (2)$$
$$\text{subject to } \ h(x) = 0 \ \text{ and } \ g(x) \le 0.$$

Definition 1. Point x^* is a CGM_c, a constrained global minimum of (P_c), if x^* is feasible and $f(x^*) \le f(x)$ for all feasible x.

When CGM_c is difficult to find, the goal of solving P_c is to find a constrained local minimum x with respect to $\mathcal{N}_c(x) = \{x' : \|x' - x\| \le \epsilon \text{ and } \epsilon \to 0\}$, the *continuous neighborhood* of x.

Definition 2. Point x^* is a CLM_c, a constrained local minimum with respect to the continuous neighborhood of x^* in P_c, if x^* is feasible and $f(x^*) \le f(x)$ for all feasible $x \in \mathcal{N}_c(x^*)$.

Based on Lagrange-multiplier vectors $\lambda = (\lambda_1, \ldots,$

$\lambda_m)^T \in \mathcal{R}^m$ and $\mu = (\mu_1, \ldots, \mu_r)^T \in \mathcal{R}^r$, the Lagrangian function of P_c is defined as:

$$L(x, \lambda, \mu) = f(x) + \lambda^T h(x) + \mu^T g(x). \quad (3)$$

a) *Karush-Kuhn-Tucker (KKT) necessary condition* [1]. Assuming x^* is a CLM_c and a regular point,[1] then there exist unique λ^* and μ^* such that:

$$\nabla_x L(x^*, \lambda^*, \mu^*) = 0, \quad (4)$$

where $\mu_j = 0 \ \forall \ j \notin A(x^*) = \{i \mid g_i(x^*) = 0\}$ (the set of active constraints), and $\mu_j > 0$ otherwise.

The unique λ and μ that satisfy (4) can be found by solving a system of nonlinear equations in λ, μ, and x. In general, (4) cannot be used effectively on multiple subproblems related by global constraints.

b) *Sufficient saddle-point condition*The concept of saddle points has been studied extensively in the past. Here, x^* is a saddle point of P_c if there exist unique λ^* and μ^* such that:

$$L(x^*, \lambda, \mu) \leq L(x^*, \lambda^*, \mu^*) \leq L(x, \lambda^*, \mu^*) \quad (5)$$

for all x that satisfies $\|x - x^*\| < \epsilon$ and all $\lambda \in \mathcal{R}^m$ and $\mu \in \mathcal{R}^r$. This condition is only sufficient but not necessary. Hence, if x^* is a CLM_c, there may not exist λ^* and μ^* that satisfy (5).

Eq. (5) is difficult to apply in practice because it is difficult to solve for unique λ^* and μ^* in a system of nonlinear inequalities.

c) *Penalty formulations.* A *static-penalty approach* [1] transforms P_c into an unconstrained minimization with the following objective function:

$$L_\rho(x, \gamma, \psi) = f(x) + \gamma^T |h(x)|^\rho + \psi^T \max(0, g(x))^\rho,$$

where $\rho > 0$. By choosing $\rho = 1$, there exist finite and sufficiently large penalty vectors $\gamma \in \mathcal{R}^m$ and $\psi \in \mathcal{R}^r$ such that x^*, a global minimum of $L_\rho(x, \gamma, \psi)$, corresponds to a *constrained global minimum* (CGM_c) of P_c. Hence, finding x^* by an unconstrained global optimization algorithm amounts to finding a CGM_c of P_c. However, the approach is hard to apply in practice because γ and ψ must be large enough in order for x^* to be a global minimum in the search space. This makes the function too rugged to be searched effectively.

A *dynamic-penalty approach* [1] increases penalties gradually and solves for the optimal solution in a sequence of unconstrained problems. However, the requirement of finding a global optimum of $L_\rho(x, \gamma, \psi)$ for each unconstrained problem may be computationally expensive in practice.

[1]Point x is a *regular point* with respect to h if gradient vectors $\nabla h_1(x), \ldots, \nabla h_m(x)$ at x are linearly independent.

2.2 Extended saddle points condition

Wah and Chen has recently proposed the theory of extended saddle points (ESP) to support constraint partitioning [26; 4]. By using a penalty function with transformed constraints, we show a necessary and sufficient condition that is satisfied for an extended region of penalty values. The latter property is important because it allows the formulation to be partitioned and used in resolving global constraints under constraint partitioning. The ESP theory is developed for general MINLPs, with CNLPs also applicable as a special case.

Consider an MINLP whose f, g and h defined in mixed space are continuous and differentiable functions with respect to the continuous variable x:

$$(P_m): \ \min_{x,y} \ f(x, y), \ \ x \in \mathcal{R}^v \text{ and } y \in \mathcal{D}^w \quad (6)$$
$$\text{subject to } \ h(x, y) = 0 \ \text{ and } \ g(x, y) \leq 0.$$

The goal of solving P_m is to find a constrained local minimum (x, y) with respect to $\mathcal{N}_m(x, y)$, the mixed neighborhood of (x, y). To define $\mathcal{N}_m(x, y)$, we need to specify its continuous and discrete counterparts. Although a continuous neighborhood is well defined, there is no accepted definition of a discrete neighborhood. We define it as follows:

Definition 3. A user-defined *discrete neighborhood* $\mathcal{N}_d(y)$ of $y \in \mathcal{D}^w$ is a *finite* user-defined set of points $\{y' \in \mathcal{D}^w\}$ in such a way that y' is reachable from y in one step, that $y' \in \mathcal{N}_d(y) \iff y \in \mathcal{N}_d(y')$, and that it is possible to reach every y'' from any y in one or more steps through neighboring points.

Intuitively, $\mathcal{N}_d(y)$ represents points that can be reached from y in one step, regardless of whether there is a valid action to effect the transition. Next, we define a mixed neighborhood and a constrained local minimum in this neighborhood:

Definition 4. A user-defined *mixed neighborhood* $\mathcal{N}_m(x, y)$ in mixed space $\mathcal{R}^v \times \mathcal{D}^w$ is:

$$\mathcal{N}_m(x, y) = \Big\{ (x', y) \mid x' \in \mathcal{N}_c(x) \Big\} \cup \Big\{ (x, y') \mid y' \in \mathcal{N}_d(y) \Big\}.$$

Definition 5. Point (x^*, y^*) is a *constrained local minimum in mixed neighborhood* (CLM_m) of P_m if (x^*, y^*) is feasible and $f(x^*, y^*) \leq f(x, y)$ for all feasible $(x, y) \in \mathcal{N}_m(x^*, y^*)$.

Before we state the main theorem, we define the ℓ-1 penalty function in a mixed space:

Definition 6. The transformed ℓ-1 penalty function of P_m in (6) is defined as follows:

$$L_m(x, y, \alpha, \beta) = f(x, y) + \alpha^T |h(x, y)| + \beta^T \max(0, g(x, y)),$$

where α and β are the *extended penalty values*.

Theorem 1. *Necessary and sufficient extended saddle point condition (ESPC) on CLM_m of P_m.* Suppose $(x^*, y^*) \in \mathcal{R}^v \times \mathcal{D}^w$ is a point in the mixed space of P_m, and the gradient vectors of the equality and the active inequality constraints with respect to x for given y^* are linearly independent. Then (x^*, y^*) is a CLM_m of P_m iff there exist finite $\alpha^* \geq 0$ and $\beta^* \geq 0$ such that, for any $\alpha^{**} > \alpha^*$ and $\beta^{**} > \beta^*$, the following condition is satisfied:

$$\begin{aligned} L_m(x^*, y^*, \alpha, \beta) &\leq L_m(x^*, y^*, \alpha^{**}, \beta^{**}) \\ &\leq L_m(x, y, \alpha^{**}, \beta^{**}) \end{aligned} \quad (7)$$

for all $(x, y) \in \mathcal{N}_m(x^*, y^*)$, $\alpha \in \mathcal{R}^m$, and $\beta \in \mathcal{R}^r$.

The following corollary facilitates the implementation of ESPC in (7) and follows directly from the definition of $\mathcal{N}_m(x, y)$. This definition allows (7) to be partitioned into two independent necessary conditions.

Corollary 1. Given $\mathcal{N}_m(x, y)$ the ESPC in (7) can be rewritten into two necessary conditions that, collectively, are necessary and sufficient:

$$\begin{aligned} L_m(x^*, y^*, \alpha, \beta) &\leq L_m(x^*, y^*, \alpha^{**}, \beta^{**}) \\ &\leq L_m(x^*, y, \alpha^{**}, \beta^{**}) \quad (8) \\ L_m(x^*, y^*, \alpha^{**}, \beta^{**}) &\leq L_m(x, y^*, \alpha^{**}, \beta^{**}) \quad (9) \end{aligned}$$

where $y \in \mathcal{N}_d((x^*, y^*) \mid x^* \text{ fixed})$ and $x \in \mathcal{N}_c((x^*, y^*) \mid y^* \text{ fixed})$.

2.3 ESPC under constraint partitioning

To solve (1) efficiently, we define a partitionable neighborhood and partition the ESPC in (7) into a set of necessary conditions that collectively are necessary and sufficient. The partitioned conditions can then be implemented by finding local saddle points in each stage of P_t and by resolving the unsatisfied global constraints using appropriate penalty values.

To enable the partitioning of ESPC into independent necessary conditions, we define the neighborhood of plan z as follows:

Definition 7. Given $\mathcal{N}_m(z(t))$, the mixed neighborhood of $z(t)$ in stage t, we define $\mathcal{N}_p(z)$, the *partitionable mixed neighborhood of plan z*, as:

$$\begin{aligned} \mathcal{N}_p(z) \;=\; \bigcup_{t=0}^{N} \mathcal{N}_p^{(t)}(z) = \bigcup_{t=0}^{N} \Big\{ z' \;\Big|\; & z'(t) \in \mathcal{N}_m(z(t)) \\ & \text{and } z'(i \mid i \neq t) = z(i) \Big\}, \end{aligned} \quad (10)$$

Intuitively, $\mathcal{N}_p(z)$ is partitioned into $N+1$ neighborhoods, each perturbing z in one of the stages of P_t. By considering P_t in (1) as an MINLP, we can apply Definition (6) and Theorem 1 to get the ESPC condition. Based on the partitionable neighborhood, our main theorem shows the partitioning of this condition into a set of distributed conditions.

Definition 8. The ℓ-1 penalty function of P_t in (1) is defined as follows:

$$\begin{aligned} L_m(z, \alpha, \beta, \gamma, \eta) = J(z) + \sum_{t=0}^{N} \Big\{ & \alpha(t)^T |h^{(t)}(z(t))| + \beta(t)^T \\ & \max(0, g^{(t)}(z(t))) \Big\} + \gamma^T |H(z)| + \eta^T \max(0, G(z)), \end{aligned} \quad (11)$$

where $\alpha(t) = (\alpha_1(t), \ldots, \alpha_{m_t}(t)) \in \mathcal{R}^{m_t}$, $\beta(t) = (\beta_1(t), \ldots, \beta_{r_t}(t)) \in \mathcal{R}^{r_t}$, $\gamma = (\gamma_1, \ldots, \gamma_p) \in \mathcal{R}^p$, and $\eta = (\eta_1, \ldots, \eta_q) \in \mathcal{R}^q$ are vectors of extended penalty values.

Theorem 2. *Distributed necessary and sufficient ESPC on CLM_m of P_t.* Plan z is a CLM_m of (1) with respect to $\mathcal{N}_p(z)$ iff the following $N+2$ conditions are satisfied:

$$\begin{aligned} \Gamma_m^{(t)}(z^*, \alpha(t), \beta(t), \gamma^{**}, \eta^{**}) &\leq \Gamma_m^{(t)}(z^*, \alpha(t)^{**}, \beta(t)^{**}, \gamma^{**}, \eta^{**}) \\ &\leq \Gamma_m^{(t)}(z, \alpha(t)^{**}, \beta(t)^{**}, \gamma^{**}, \eta^{**}), \\ L_m(z^*, \alpha^{**}, \beta^{**}, \gamma, \eta) &\leq L_m(z^*, \alpha^{**}, \beta^{**}, \gamma^{**}, \eta^{**}), \end{aligned}$$

for all $z \in \mathcal{N}_p^{(t)}(z^*)$ and $\alpha(t) \in \mathcal{R}^{m_t}$, $\beta(t) \in \mathcal{R}^{r_t}$, $\gamma \in \mathcal{R}^p$, $\eta \in \mathcal{R}^q$, and $t = 0, \ldots, N$, where

$$\begin{aligned} \Gamma_m^{(t)}(z, \alpha(t), \beta(t), \gamma, \eta) = J(z) + \alpha(t)^T |h^{(t)}(z(t))| + \beta(t)^T \\ \max(0, g^{(t)}(z(t))) + \gamma^T |H(z)| + \eta^T \max(0, G(z)). \end{aligned}$$

By using a partitionable neighborhood, Theorem 2 shows that the original ESPC in Theorem 1 can be **partitioned into $N+1$ necessary conditions**, each of which corresponds to finding a saddle point in a stage of the original problem. Hence, the original problem is now reduced to solving multiple smaller subproblems and to the resolution of unsatisfied global constraints across the subproblems. By reducing the solution space in each subproblem through the search of saddle points, Theorem 2 leads to a significant reduction in the base of the exponential complexity in finding CLM_m.

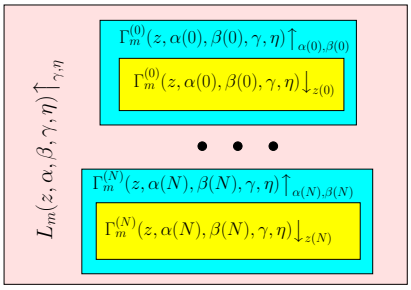
An important aspect of Theorem 1 over the original saddle-point condition in (5) is that, instead of solving a system of nonlinear equations to find unique λ^* and μ^* that minimize $L(x, \lambda^*, \mu^*)$ at x^*, it suffices to find any $\alpha^{**} > \alpha^*$ and $\beta^{**} > \beta^*$. Such a property allows the solution of P_m to be implemented iteratively by looking for a local minimum (x^*, y^*) of $L_m(x, y, \alpha, \beta)$ with respect to points in $\mathcal{N}_m(x^*, y^*)$ in an inner loop, and for any $\alpha^{**} > \alpha^*$ and $\beta^{**} > \beta^*$ in an outer loop.

Figure 1a shows the pseudo code that implements the conditions in Corollary 1. The two inner loops look for local minima of $L_m(x, \alpha, \beta)$ in the continuous and discrete neighborhoods, whereas the outer loop performs ascents on α and β for unsatisfied global constraints and stops when a CLM_m has been found.

The iterative search can be extended to the distributed conditions in Theorem 2. In Figure 1b, the two inner nested loops of stage t look for a local saddle point of

$\alpha \longrightarrow 0$; $\beta \longrightarrow 0$;
repeat
  increase α_i by δ_i if $h_i(x,y) \neq 0$ for all i;
  increase β_j by δ_j if $g_j(x,y) \not\leq 0$ for all j;
  repeat
    perform descent of $L_m(x,y,\alpha,\beta)$ with respect to x for given y;
  until a local minimum of $L_m(x,y,\alpha,\beta)$ with respect to x for given y has been found;
  repeat
    perform descent of $L_m(x,y,\alpha,\beta)$ with respect to y for given x;
  until a local minimum of $L_m(x,y,\alpha,\beta)$ with respect to y for given x has been found;
until a CLM_m of P_m has been found or ($\alpha > \bar{\alpha}^*$ and $\beta > \bar{\beta}^*$);

a) Implementation of Theorem 1



b) Implementation of Theorem 2

Figure 1: Simple iterative implementation of ESPC to look for CLM_m of P_m and that of distributed ESPC to look for CLM_m of P_t.

$\Gamma_m^{(t)}(z,\alpha(t),\beta(t),\gamma,\eta)$. This is done by updating z and $\alpha(t)$ and $\beta(t)$ associated with the local constraints, using fixed γ and η associated with the global constraints. With fixed γ and η, the algorithm is actually finding $z(t)$ that solves the following MINLP in stage t:

$$\min_{z(t)} \quad J(z) + \gamma^T H(z) + \eta^T G(z) \qquad (12)$$
$$\text{subject to} \quad h^{(t)}(z(t)) = 0 \quad \text{and} \quad g^{(t)}(z(t)) \leq 0.$$

Since this is a well-defined MINLP, any existing solver with little modification can be used to solve it. We have studied this approach in discrete planning domains by using ASPEN to solve subproblems partitioned by a discrete version of Theorem 2 [6].

After performing the local searches, the penalties on unsatisfied global constraints are increased in the outer loop. The search iterates until a feasible local minimum in the constrained model has been found.

3 Continuous and Differentiable Nonlinear Optimization Formulation

In this section, we present our CNLP formulation for temporal planning problems. We make several remarks before presenting the formulation. First, our formulation is based on a partial order casual link (POCL) paradigm that has been used for SAT encoding [16] and constrained programming encoding [25] before. Second, the CNLP formulation we developed is limited to canonical solution plans [12] in which every grounded action can be executed at most once. It is known that for most benchmarks in temporal planning, the optimal plans are canonical [25]. It is possible to extend a canonical planner to a complete planner by duplicating actions.

A temporal planning problem is briefly defined as follows.

Definition 1 (State) *Given a set of atomic facts $F = \{f_1, f_2, \cdots, f_n\}$, a state s is a subset of F.*

Definition 2 (Action) *An action o is a triple o=(pre(o), add(o), del(o)), where pre(o)$\subseteq F$ is the preconditions of o, add(o)$\subseteq F$ is the set of added facts of o, and del(o)$\subseteq F$ is the set of deleted facts of o*[2]*.*

The result of applying a single action o to a state s is defined as follows,

$$Result(s,o) = \begin{cases} (s \vee add(o) \setminus del(o)), & \text{if } pre(o) \subseteq s; \\ undefined, & \text{otherwise.} \end{cases}$$

From above definition, for a sequence $P = \langle o_1, o_2, \cdots, o_n \rangle$ of actions, we have $Result(s,P) = Result(Result(Result(s,o_1),o_2)\cdots,o_n)$.

Definition 3 (Planning task) *A planning task is a tuple $T = (O,F,I,G)$, where O is a set of actions, F is a set of facts, $I \subseteq F$ is the initial state, and $G \subseteq F$ is the goal state.*

3.1 Logical encoding

In our CNLP formulation, we use an action-based encoding in which all variables are associated with the time assignments of actions. Before presenting the CNLP formulation, we first less rigorously describe a constraint programming formulation containing logical operators OR ($\vee$) and AND ($\wedge$). We then transform it into a CNLP formulation.

Given a planning task $T = (O,F,I,G)$, we assign each action $a \in O$ a **starting time** $s(a)$ and denote its **duration** as $t(a)$. In addition, we create two artificial actions representing the initial and goal states. For the initial state, we add an action a_s with zero precondition and delete-effects , with the facts in I as add effects, and with zero duration:

$$pre(a_s) = del(a_s) = \emptyset,\ add(a_s) = I,\ t(a_s) = 0.$$

For the goal state, we add an action a_g with the facts in G as preconditions, with all facts as add-effects, and with zero duration:

$$pre(a_g) = G, add(a_g) = F, del(a_g) = \emptyset,\ t(a_g) = 0.$$

[2] In full PDDL2.1 specification, preconditions and effects can be effective at different times in the lifetime of an action. We have adopted a simplified version here.

We add these two actions into the original set of actions to define an extended set of action:

$$O^+ = O \cup \{a_s, a_g\},$$

and we fix the starting time of the initial action to zero:

$$s(a_s) = 0.$$

A partial order casual link STRIPS plan is an assignment of the starting time of actions satisfying the following conditions:

1) Every prerequisite has a cause. If an action is assigned at time t, then for each of the preconditions of a, there is at least one action before a supporting the precondition.

$$\bigvee_{b \in sup(f)} \Big(s(b) + t(b) < s(a) \Big)$$
$$\forall a \in O^+, \ \forall f \in pre(a) \tag{13}$$

where $\vee$ means logically OR, $sup(f)$ defines the set of actions that support a fact f as an add-effect:

$$sup(f) = \{a | f \in add(a), a \in O^+\}.$$

Note that since the goal action a_g supports all facts as add-effects, a_g is in $sup(f)$ for the all facts. The implicit effect of this scheme is that if an action happens after a_g, it is in fact an *inactive action* whose preconditions do not need to be supported by regular actions. Indeed, in the final solution to this encoding, we treat all actions after a_g as unused actions and do include them in the solution plan.

2) Every prerequisite support is maintained. For every prerequisite fact f of an action a, for each action c that invalidates f, either c happens after a, or there is another action b that supports f and happens between c and a.

$$\bigvee_{b \in sup(f)} \Big(s(b) + t(b) < s(a) \bigwedge s(c) + t(c) < s(b) \Big)$$
$$\bigvee \Big(s(a) + t(a) < s(c) \Big) \bigvee \Big(s(a_g) < s(a) \Big),$$
$$\forall a \in O^+, \ \forall f \in pre(a), \ \forall c \in inv(f), \tag{14}$$

where $inv(f)$ defines the set of actions that invalidate a fact f as a delete-effect:

$$inv(f) = \{a | f \in del(a), a \in O^+\}.$$

The last part of the above function, $s(a_g) < s(a)$, ensures that if an action is an inactive action (happening after a_g), all the other requirements can be waived.

The constraints in (13) and (14) ensure that the solution plan is a valid temporal plan. Note that the information of the initial and goal states are implicitly encoded using the special a_s and a_g actions, which are included in O^+. All the subgoals in G will be reached because they are preconditions of a_g and they will need to be supported.

3.2 CNLP transformation

Constraint functions in the previous logical encoding are not continuous or differentiable and contain logical operators that cannot be handled by CNLP solvers. We derive a continuous and differentiable CNLP formulation from the above logical encoding as follows:

(CNLP-PLANNING):

$$\min \quad J = s(a_g) \tag{15}$$

$$\text{s.t.} \quad \sum_{b \in sup(f)} \mu_{a,b}(s(b) + t(b) - s(a)) < 0,$$
$$\forall a \in O^+, \ \forall f \in pre(a) \tag{16}$$

$$\text{and} \quad \sum_{b \in sup(f)} \theta_{a,b,c}(s(b) + t(b) - s(a))(s(b) - s(c) - t(c))$$
$$+\xi_{a,c}(s(a) + t(a) - s(c)) + (s(a_g) - s(a)) < 0,$$
$$\forall a \in O^+, \ \forall f \in pre(a), \ \forall c \in inv(f). \tag{17}$$

The variables in this encoding include: $s(a) \in [0, T_{max}]$ for all $a \in O^+$ representing the starting time of actions; and $\mu_{a,b} > 0$, $\theta_{a,b,c} > 0$, and $\xi_{a,c} > 0$ which are positive auxiluary variables. The obejctive function $s(a_g)$ enforces the minimization of the makespan of the plan. Changing the objective function J provides a flexible way to specify other plan quality metrics.

The CNLP-PLANNING formulation is a direct compliation of the logical encoding in Section 3.1. Each constraint in (13) requires that at least one action b supporting the precondition fact f is before a. We specify this condition in (16) in the CNLP model. In (16), since μ are all positive, it can be satisfied only when there is at least one action b in $sup(f)$ before a ($s(b) + t(b) - s(a) < 0$).

Each constraint in (14) requires that a is after a_g, or the invalidating action c is after a, or at least one action b supporting the precondition fact f is after c and before a (to recover f). We specify this condition in (17) in the CNLP model. In (17), since θ and ξ are all positive, it can be satisfied only when

- a is after a_g ($s(a_g) - s(a) < 0$),
- *or* the invalidating action c is after a ($s(a) + t(a) - s(c) < 0$),
- *or* there is at least one action $b \in sup(f)$ between c and a ($(s(b) + t(b) - s(a))(s(b) - s(c) - t(c)) < 0$).

Proposition 1. A feasible solution to the CNLP-PLANNING problem corresponds to a feasible temporal plan of the original problem.

Proposition 2. The optimal CGM_c solution to the CNLP-PLANNING problem corresponds to an optimal temporal plan of the original planning problem under the quality metric specified in the objective function J.

It should be noted that the CNLP formulation incorporates the full durations of actions in the constraint func-

tions and thus prohibits parallel execution of mutual exclusive active[3] actions. In fact, for each pair of active actions a and b that are mutual exclusive , the duration of a and b cannot overlap, because the formulation requires that $s(a) + t(a) < s(b)$ or $s(b) + t(b) < s(a)$.

The CNLP formulation is a continuous and differentiable problem that can be solved by any continuous nonlinear programming packages. Since for all actions, the number of preconditions and number of invalidating actions are typically bounded by a small constant, the number of constraints is of the order $O(|O|)$, where $|O|$ is the number of actions. The number of variables is also of the order of $O(|O|)$. Note that although the auxiliary variables $\mu_{a,b}$ is indexed by actions a and b, only certain combinations where b is a supporting action for a are needed. Therefore each action a is associated only with a small number of $\mu_{a,b}$. Similar observations can be made for θ and ξ.

4 Constrained Partitioning on the CNLP Formulation

It is straightforward to apply the constraint partitioning approach and the ESP theory discussed in Section 2 to the CNLP formulation. Like SGPlan, we partition the constraints by subgoals. The constraints related to the the goal action a_g are treated as global constraints. Specifically, since $pre(a_g) = G$, where G is the set of subgoals, for each subgoal $f \in pre(a_g)$, there is a group of constraints in (CNLP-PLANNING) enforcing the support of f. Since in a partitioned search, each subproblem only solves for one subgoal f, the group of constraints supporting f are treated as local constraints, and other constraints for supporting other subgoals in $pre(a_g)$ are treated as global constraints.

We have developed CNLP-SGPlan, an implementation of the partitioned search procedure in Figure.1 on the CNLP formulation. For each subgoal f, we solve a local CNLP problem involving local constraints for the satisfaction of the subgoal f and a biased objective function incorporating the violation of global constraints. Note that although in the ESP theory, the biased objective function contains $|\cdot|$ and $max(\cdot)$ functions that are not differentiable, we can apply a simple transformation to transform each local subproblem into a CNLP with a continuous and differentiable objective function [4].

5 Preliminary Evaluation

We have performed preliminary study of the performance of CNLP-SGPlan on some planning benchmarks. We use the preprocessing part of SGPlan [5], which is based on a modified parser of Metric-FF [15], to transform an input planning problem into the CNLP formulation. The output is written in the AMPL optimazation modelling format [10]. Since the evaluation version of AMPL environment we have now can only handle problems with less than 300 variables and 300 constraints, we cannot run the software locally as most CNLP-PLANNING problems obtained from planning benchmarks exceed the size limit. Instead, We submit each AMPL problem to the NEOS optimization server [20] and select a CNLP solver to solve it.

In our experiments, we have first tried several smaller domains from the Third International Planning Competition (IPC3), including Depots and Driverlog, to find out a suitable nonlinear optimization package for the CNLP problems derived from planning applications.

We have tested several leading CNLP packages including: SNOPT [14], a large-scale optimization package implementing a trust-region sequential quadric programming (SQP) method [1]; Lancelot [7], a dynamic penalty method based on an augmented Lagrangian formulation; and PENNON [18], a generalized augmented Lagrangian method. We have found that PENNON significantly outperforms other two candidates in terms of solution speed on the planning-based CNLP problems. The solution plans found by CNLP-SGPlan have consistently better makespan than the solutions found by SGPlan, which demonstrates the effectiveness of the objective function in the CNLP-PLANNING formulation.

We then apply CNLP-SGPlan, with PENNON as the basic CNLP solver, to the AIRPORT domain from IPC4 [9]. In contrast to the original SGPlan planner that can only solve the up to the 46^{th} problem instance (out of 50 instances), CNLP-SGPlan is able to solve the largest problems numbered from 47 to 50 in less than 1800 seconds. With an MINLP formulation, each subproblem in the original SGPlan usually requires several minutes to solve for the largest instances. In contrast, the time PENNON takes to solve CNLP subproblems is of the order of ten seconds for the largest instances. We are in the process of obtaining a full AMPL preprocessor and performing complete evaluation of CNLP-SGPlan in the future.

6 Related Work and Discussions

Transformation is a popular approach to planning. Transformation-based planners include those planners that transform a planning problem into another formulation before solving it. Example formulations include SAT used by SATPLAN and Blackbox [23], integer linear programming models used by ILP-Plan [17], constrained programming models used by CPT [25], and the mixed-integer nonlinear programming model used by SGPlan [5]. All these models have discrete or mixed-integer variables, and the functions in these models are not continuous and differentiable. Among all the formulations, the CNLP formulation proposed in this paper is generally much more efficient to solve than others because the CNLP solvers can utilize the powerful gradient

[3] An action a is active when $s(a) < s(a_g)$.

information provided by the CNLP functions.

Systematic searches explore the entire state space completely. Examples include UCPOP [21], Graphplan [2], STAN [19], and PropPLAN [11]. These methods are in general very expensive.

Heuristic solvers prioritize the search space by heuristic functions. Examples include HSP [3], Metric-FF [15], GRT [22], MIPS [8], and Sapa [24]. There are also local search solvers such as LPG [13]. Heuristic search and local search in general do not guaranteed to find feasible plans and do not have any degree of guarantee on the solution quality.

7 Conclusions

In this paper, we have proposed a continuous and differentiable nonlinear optimization formulation for optimal temporal planning. By providing gradient information through the CNLP formulation, the transformed problem can be solved efficiently and optimally by sophisticated CNLP solvers. We have proposed to further improve the efficiency by applying the constraint partitioning approach to the CNLP formulation and solve the problem in a partitioned fashion based on the recently proposed theory of extended saddle points.

In the future, we plan to study the extension of the model to numerical domains, investigate the reduction of encoding size, and systematically evaluate the CNLP formulation in various application domains.

References

[1] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, Massachusetts, 1999.

[2] A. L. Blum and M. L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90:281–300, 1997.

[3] B. Bonet and H. Geffner. Planning as heuristic search. *Artificial Intelligence, Special issue on Heuristic Search*, 129(1), 2001.

[4] Y. Chen. *Solving Nonlinear Constrained Optimization Problems Through Constraint Partitioning*. PhD thesis, University of Illinois at Urbana-Champaign, 2005.

[5] Y. Chen, C. Hsu, and B. W. Wah. Sgplan: Subgoal partitioning and resolution in planning. In *Proc. IPC4, ICAPS*, pages 30–32, 2004.

[6] Y. Chen and B. W. Wah. Automated planning and scheduling using calculus of variations in discrete space. In *Proc. Int'l Conf. on Automated Planning and Scheduling*, pages 2–11, June 2003.

[7] A. R. Conn, N. Gould, and Ph. L. Toint. Numerical experiments with the LANCELOT package (Release A) for large-scale nonlinear optimization. *Mathematical Programming*, 73:73–110, 1996.

[8] S. Edelkamp. Mixed propositional and numerical planning in the model checking integrated planning system. In *Proc. Workshop on Planning for Temporal Domains*. AIPS, 2002.

[9] S. Edelkamp and J. Hoffmann. Classical part, 4th international planning competition. http://ls5-www.cs.uni-dortmund.de/ edelkamp/ipc-4/, 2004.

[10] R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Brooks Cole Publishing Company, 2002.

[11] M. P. Fourman. Propositional planning. In *Proc. Workshop on Model Theoretic Approaches to Planning*. AIPS, 2000.

[12] H. Geffner. Planning as branch and bound and its relation to constraint-based approaches. In *Technical report, Universidad Simon Bolivar*, 2001.

[13] A. Gerevini and I. Serina. LPG: a planner based on local search for planning graphs with action costs. In *Proc. of the Sixth Int. Conf. on AI Planning and Scheduling*, pages 12–22. Morgan Kaufman, 2002.

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

[15] Jörg Hoffmann. The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research*, 20:291–341, 2003.

[16] H. Kautz, D. McAllester, and B. Selman. Encoding plans in propositional logic. In *Proceedings of KR-97*, pages 374–384, 1996.

[17] H. Kautz and J. P. Walser. Integer optimization models of AI planning problems. *The Knowledge Engineering Review*, 15(1):101–117, 2000.

[18] M. Kocvara and M. Stingl. Pennon: A code for convex nonlinear and semidefinite programming. *Optimization Methods and Software*, 18(3):317–333, 2003.

[19] D. Long and M. Fox. Efficient implementation of the plan graph in STAN. *J. of AI Research*, 1998.

[20] NEOS. The NEOS server for optimization. *http://www-neos.mcs.anl.gov/neos/*, 2005.

[21] J. Penberethy and D. Weld. UCPOP: A sound, complete, partial order planner for ADL. In *Proc. 3rd Int'l Conf. on Principles of Knowledge Representation and Reasoning*, pages 103–114. KR Inc., 1992.

[22] I. Refanidis and I. Vlahavas. The GRT planner. *AI Magazine*, pages 63–66, 2001.

[23] B. Selman and H. Kautz. Planning as satisfiability. In *Proceedings ECAI-92*, pages 359–363, 1992.

[24] M. B. D. Subbarao and S. Kambhampati. Sapa: A domain-independent heuristic metric temporal planner. Technical report, Arizona State University, 2002.

[25] V. Vidal and H. Geffner. "branching and pruning: An optimal temporal pocl planner based on constraint programming". In *Proc. AAAI-04*, 2004.

[26] B. W. Wah and Y. Chen. Constraint partitioning in penalty formulations for solving temporal planning problems. *Artificial Intelligence*, (accepted for publication), 2006.