Automated Planning and Scheduling using Calculus of Variations in Discrete Space*

Yixin Chen and Benjamin W. Wah
Department of Electrical and Computer Engineering
and the Coordinated Science Laboratory
University of Illinois, Urbana-Champaign
Urbana, IL 61801, USA
E-mail: {chen, wah}@manip.crhc.uiuc.edu
URL: http://manip.crhc.uiuc.edu

Abstract

In this paper, we propose new dominance relations that can speed up significantly the solution process of planning problems formulated as nonlinear constrained dynamic optimization in discrete time and space. We first show that path dominance in dynamic programming cannot be applied when there are general constraints that span across multiple stages, and that node dominance, in the form of Euler-Lagrange conditions developed in optimal control theory in continuous space, cannot be extended to that in discrete space. This paper is the first to propose efficient node-dominance relations, in the form of local saddle-point conditions in each stage of a discrete-space planning problem, for pruning states that will not lead to locally optimal paths. By utilizing these dominance relations, we present efficient search algorithms whose complexity, despite exponential, has a much smaller base as compared to that without using the relations. Finally, we demonstrate the performance of our approach by integrating it in the ASPEN planner and show significant improvements in CPU time and solution quality on some spacecraft scheduling and planning benchmarks.

Introduction

Many planning, scheduling and control applications can be formulated naturally as constrained *dynamic optimization problems* with dynamic variables that evolve over time. These problems can generally be classified into four types:

- continuous-time continuous-state problems,

- discrete-time continuous-state problems,

- continuous-time discrete-state problems, and

- discrete-time discrete-state problems.

The first two classes have been studied as *functional optimization problems* in classical calculus of variations and as *optimal control problems* in control theory. In these two classes of problems, variational techniques and calculus of variations are the major solution tools when the objective and constraint functions are continuous and differentiable.

In this paper, we are interested in the last class defined above, namely, planning problems formulated as *nonlinear constrained dynamic optimization problems in discrete space and time*:

$$\min_{y} \quad J[y] = \sum_{t=0}^{N} F(t, y(t), y(t+1)) \qquad (1)$$

$$\text{subject to} \quad E(t, y(t)) = 0, \quad t = 0, \cdots, N+1,$$
$$G(t, y(t), y(t+1)) = 0,$$
$$\text{and} \quad I(y) = 0,$$

with dummy constraints $G(N+1, y(N+1), y(N+2)) = 0$ always satisfied. Here, $y_i(t)$ is the ith discrete *dynamic state variable* in stage (time step) t; $y(t) = (y_1(t), \cdots, y_u(t))^T$ is a u-component *state vector* in discrete space $\mathcal{Y}$; $E = (E_1, \cdots, E_r)^T$ is a r-component vector of functions called the local constraints; $G = (G_1, \cdots, G_p)^T$ is a p-component vector of *Lagrange constraints* (Cadzow 1970); and $I = (I_1, \cdots, I_q)^T$ is a q-component vector of *general constraints*. In this definition, a local constraint only involves local state variables in one stage, a Lagrange constraint involves state variables in two adjacent stages, and a general constraint involves state variables across more than two stages. Note that F, G and I are *not* necessarily continuous and differentiable. For simplicity and to mirror the defintion in traditional calculus of variations, we have defined an objective that is composed of multiple objectives, each across a pair of stages. However, this assumption can be relaxed by using a general objective function in the same way as a general constraint in the formulation. Lastly, although we have defined (1) with equality constraints, we show extensions later for handling inequality constraints.

A solution $y = (y(0), y(1), \cdots, y(N+1))^T$ to (1) consists of u discrete-time curves,[1] one for each dynamic state variable. Following conventional terminologies in continuous control theory, we call y a *bundle* (or a vector of curves) for both continuous- and discrete-time cases, and $J[y]$, a *functional* defined as a mapping from y to a value in R.

Many applications in discrete time and space formulated in (1) are characterized by multiple stages (discrete time horizons) with a discrete search space, local constraints

[1]In this paper, we use the term "curve" for both continuous- and discrete-time cases. A discrete-time curve is typically named a "sequence."
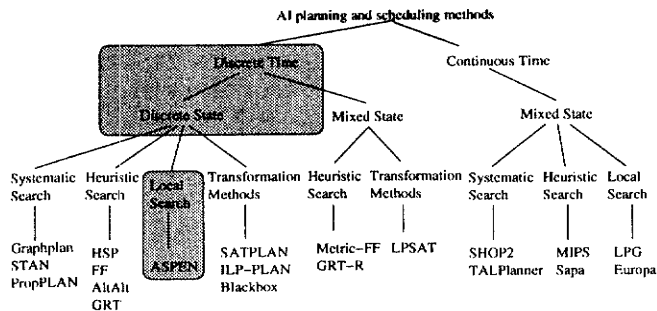
Figure 1: A classification of existing approaches, where the shaded boxes indicate our focus in this paper.

a) Principle of Optimality applied to feasible states

b) Termination of c_1 by node dominance, independent of P_1
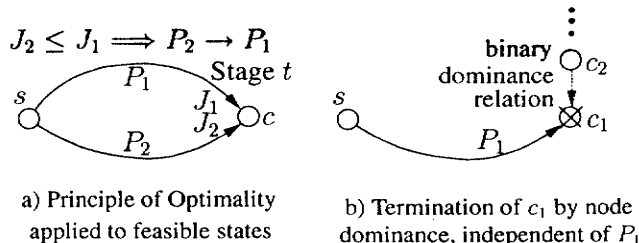
Figure 2: The application of path and node dominance in constrained minimization.

Table 1: Worst-case complexities of enumeration algorithms with path and/or node dominance, where $|s| \leq |\mathcal{Y}|$.

Constraint Type	Without General Constraints		With General Const. (W/O Path Dominance)	
Dominance Used	With Path Dominance	With Path & Node Dom.	W/O Node Dominance	With Node Dominance
Complexity	$O\ N|\mathcal{Y}|^2$	$O\ N(|\mathcal{Y}| + |s|^2)$	$O\ |\mathcal{Y}|^N$	$O\ |\mathcal{Y}|N|s|^N$

in each stage, Lagrange constraints between neighboring stages, general constraints across more that two stages, and procedural objective and constraint functions. Ample examples exist in production planning and scheduling, chemical control processing, automated vehicle path planning, and action scheduling. For example, some AI planning problems can be formulated in (1) using a discrete planning horizon, discrete state vectors representing positive and negative facts in each stage, and constraints representing preconditions and effects of actions. In that case, both space and time are discrete. Another example is in planning spacecraft operations over a horizon. The application involves finding a sequence of low-level commands from a set of high-level science and engineering goals, such as spacecraft operability constraints, flight rules, hardware models, science experiment goals, and operation procedures, subject to parameter dependencies and temporal and resource constraints. In this application, a Lagrange constraint relates state variables in adjacent time steps, whereas a general constraint may represent the total fuel available in the horizon.

Figure 1 classifies existing AI planning and scheduling methods and our focus in this paper.

Existing methods for solving discrete-state discrete-time problems can be classified into four categories:

- Systematic search methods that explore the entire state space are expensive because they are enumerative. Examples include Graphplan, STAN and PropPLAN.

- Heuristically guided and local searches that search in discrete path space depend heavily on the guidance heuristics used and are not guaranteed to find feasible bundles. Examples include HSP, FF, AltAlt, and ASPEN.

- Transformation methods transform a planning problem into a constrained optimization or satisfaction problem before solving it by existing constrained programming techniques. They allow objectives to be coded easily and discrete resource constraints to be handled. However, constrained searches are too computationally expensive when applied to solve large planning problems. Examples include SATPLAN, ILP-PLAN, and Blackbox.

To overcome the exponential complexity of systematic searches, it is essential to develop techniques for reducing the worst-case exponential complexity of these methods. In this paper, we propose powerful node-dominance relations

in order to reduce the base of the exponential complexity.

The class of discrete-time, mixed-state problems have been solved by heuristic methods (Metric-FF, GRT-R) and transformation methods (LPSAT), whereas the class of continuous-time, mixed-state problems have been solved by systematic searches (SHOP2, TALplanner), heuristic methods (MIPS, Sapa), and local-search methods (LPG, Europa). We discuss later the extension of our proposed approach to these two classes.

In general, the complexity of a multi-stage path-search problem can be reduced by dominance relations. Such relations can be classified into path dominance studied in traditional dynamic programming and node dominance.

A special case of (1) with local and Lagrange constraints but without general constraints can be solved by dynamic programming, as is illustrated on an intermediate feasible state c in Figure 2a. The *Principle of Optimality* (Bellman 1957) states that, if c lies on the optimal bundle from s to the final state, then it is necessary and sufficient for the bundle from s to c to be optimal. According to the principle, if bundle P_1 (*resp.* P_2) from s to c has objective value J_1 (*resp.* J_2), based on variables assigned in P_1 (*resp.* P_2), and $J_2 \leq J_1$, then P_1 cannot to be part of the optimal bundle. That is, P_1 is *path dominated* by P_2 ($P_2 \rightarrow P_1$).

Since dominating paths involve only feasible states in each stage and there is no efficient algorithm for identifying feasible states when constraints are nonlinear, we may need to enumerate all possible states in each stage in the worst case. To estimate this worst-case complexity, consider a simple $(N + 2)$-stage search problem, with an initial state in the first stage, a final state in the last stage, and $|\mathcal{Y}|$ states in stage $t = 1, \cdots, N$, where $\mathcal{Y}$ is the discrete search space in each stage. The complexity to find an optimal bundle is $O\left(N|\mathcal{Y}|^2\right)$, which is polynomial in $|\mathcal{Y}|$. The second column in Table 1 summarizes this complexity.

When general constraints $I(y) = 0$ are present in (1), the Principle of Optimality is not satisfied because a path-dominated partial bundle in a stage may satisfy a general constraint that spans beyond this stage, whereas a path-
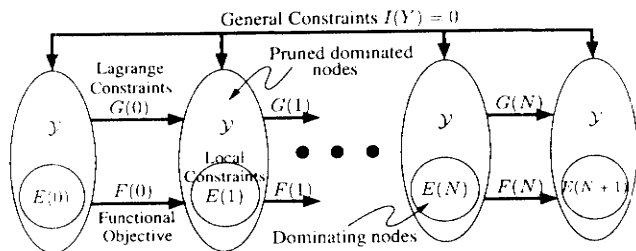
Figure 3: The pruning of dominated states in each stage leads to a smaller search space in finding optimal bundles across stages.

dominating partial bundle may not. Hence, an algorithm for finding an optimal bundle may need to enumerate all possible bundles across all stages in the worst case, leading to a complexity exponential[2] in $|\mathcal{Y}|$ (fourth column in Table 1).

Another type of dominance that does not involve the Principle of Optimality is a binary node-dominance relation $c_2 \rightarrow c_1$ that prunes c_1 in Figure 2b. Node dominance relations are conditions that relate one state to another in such a way that a dominated state can be pruned safely because it cannot lead to better feasible objective values than those of a dominating state. They are different from path dominance because they only depend on local state variables and not on paths leading to these states. They help reduce the overall number of bundles enumerated because any feasible bundle should involve only dominating but not dominated states in each stage. Note that node dominance is only necessary for global optimality in (1).

When only local and Lagrange constraints are present in (1), node dominance, if exists, can be combined with path dominance to further reduce the search complexity. For simplicity, let $s \subseteq \mathcal{Y}$ be the set of dominating states not pruned by node dominance in stage $t = 0, \cdots, N + 1$ in the multi-stage example above. An exhaustive search for finding an optimal bundle will first enumerate all states in stage t and then apply node dominance in order to identify s. It then applies path dominance on the $|s|^2$ pairs of states across adjacent stages in order to identify the optimal bundle. This leads to a worst-case complexity of $O\left(N(|\mathcal{Y}| + |s|^2)\right)$, which is better than that when path dominance is applied alone (third column in Table 1).

Last, when general constraints $I(y) = 0$ are present in (1), only node dominance but not path dominance can be applied. Node dominance in this case helps restrict the state space in each stage to be searched. For instance, an algorithm for finding an optimal bundle may enumerate all possible combinations of bundles of $|s|$ states in stage $t = 0, \cdots, N + 1$, leading to a worst-case complexity of $O\left(N|\mathcal{Y}| \times |s|^N\right)$ (last column in Table 1). Here, we assume a worst-case complexity of $|\mathcal{Y}|$ in stage t for finding a dominating state in s. Since $|s|$ is generally much smaller than $|\mathcal{Y}|$, node dominance helps reduce the base of the worst-case exponential complexity to a much smaller value.

---

[2] The implicit assumption is that (1) with general nonlinear discrete constraints is NP-hard; hence, it is unlikely that the problem is polynomially solvable. An enumerative algorithm can find an optimal bundle in worst-case complexity $O\left(|\mathcal{Y}|^N\right)$.

Figure 3 illustrates the reduction in search space due to node dominance. By partitioning the search into stages and by finding only dominating states in each stage, the search can be restricted to only dominating states in each stage. This focus will lead to a significant reduction on search complexity, especially when it takes exponential time to find optimal bundles in the presence of general constraints.

In the calculus of variations in classical control theory, node dominance in Figure 2b has been applied to solve continuous-state problems. Since the conditions in continuous space are based on necessary conditions in the theory of Lagrange multipliers, they can only be proved to be necessary. As a result, a dominating state may not always lead to feasible bundles, even when general constraints are absent.

We have introduced new node-dominance relations for solving (1) (Chen & Wah 2002) by partitioning global necessary conditions, based on the theory of Lagrange multipliers in discrete space (Wah & Wu 1999), into multiple sets of local node-dominance relations, one for each stage of the problem. The use of Lagrange multipliers in the global problem as well as the decomposed subproblem in each stage allows these problems to be solved in a uniform fashion.

Our approach extends the calculus of variations in continuous space to that in discrete space. We first formulate (1) without general constraints in a Lagrangian function with discrete variables. Based on the theory of Lagrange multipliers in discrete space (Wah & Wu 1999), we reduce the search to finding a saddle point in a discrete neighborhood of the bundle, where a neighborhood is the union of discrete neighborhoods in each stage. Since such a neighborhood is very large when the number of stages is large, it is hard to find feasible bundles within it. To this end, we partition the original Lagrangian function into multiple Lagrangian functions, each involving only local variables in a stage of the problem. The search for a saddle point in the original neighborhood is then reduced to the search for multiple local saddle points, one in each stage. Moreover, we show that the collection of local saddle points are necessary for local optimality.

To implement the theory, we present DCV+CSA, a general search algorithm that uses constrained simulated annealing (CSA) (Wah & Wang 1999) to look for local saddle points in each stage, and the dynamic decomposition of time steps into stages. We employ CSA (Wah & Wang 1999) to look for saddle points in each stage by performing descents in the original variable subspace and ascents in the Lagrange-multiplier subspace. To demonstrate our approach, we integrate DCV+CSA into ASPEN (Chien, *et al.* 2000) and show significant improvements in time and quality on some spacecraft planning benchmarks, as compared to those obtained by ASPEN.

Figure 4 illustrates our approach on a toy example in ASPEN (Chien, *et al.* 2000). The application involves scheduling four activities over a horizon of 60 seconds, while satisfying various constraints relating the activities and "power_resource," "color_state," and "color_changer." By partitioning the horizon into three stages, eleven constraints are localized, leaving only two Lagrange constraints and two general constraints. In this toy example, the number

```
model toy {HORIZON_START = 1998-1/00:00:00; horizon_duration = 60s; time_scale = second;};
parameter string color {domain = ("red", "blue", "green");};
State_variable color_sv {states = ("red", "blue", "green"); default_state = "red";};
Resource power {type = non_depletable; capacity = 25; min_value = 0;};
Activity color_changer {color c; duration = 1; reservations = color_sv change_to c;};
Activity A1 {duration = [10,20]; constraints = ends_before start of A2 by [0,30];
    reservations = power use 10, color_sv must_be "green";};
Activity A2 {duration = 10; reservations = power use 20, color_sv must_be "blue";};
// initial schedule
A1 act_1 {start_time = 0; duration = 15;};    act_2 { start_time = 20; duration = 10;};
A2 act_3 {start_time = 30; duration = 10;};   act_4 { start_time = 50; duration = 10;};
```
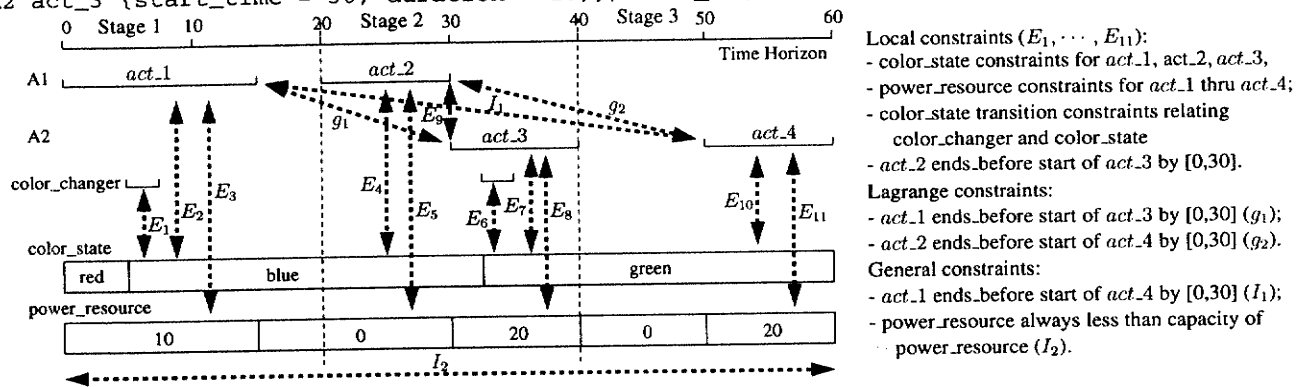

Figure 4: A toy example from ASPEN (Chien, *et al.* 2000) in which the goal is to find a valid schedule that completes activities act_1, act_2, act_3, and act_4. The horizon is divided into three stages: $[0,20]$, $[21,40]$, $[41,60]$.

of iterations is reduced from 16 taken by ASPEN to 12 taken by our proposed ASPEN+DCV+CSA.

Previous Work

Calculus of variations in continuous space

In classical variational theory, the simplest form of a fundamental continuous-time continuous-state functional optimization problem is defined as follows:

$$\min_{y} J[y] = \int_{t_0}^{t_1} F(t, y(t), y'(t))dt, \quad \text{where } y(t) \in R^u. \quad (2)$$

Since it is generally difficult to find conditions for an absolute minimum curve in a search space, the theory in continuous space defines the concepts of *strong* and *weak relative minimum* (Dreyfus 1965). The study led to a number of necessary conditions for a curve to be a weak relative minimum, among which the most important is the *Euler-Lagrange* condition stated below in its differentiated form.

Theorem 1. *Continuous-time continuous-state Euler-Lagrange condition* (Dreyfus 1965). If bundle y is a weak relative minimum to (2), and if t is any point in $[t_0, t_1]$ where derivative $y'(t)$ exists, then the following Euler-Lagrange equation holds:

$$F_{y_i}(t, y(t), y'(t)) - \frac{\partial}{\partial t} F_{y_i'}(t, y(t), y'(t)) = 0. \quad (3)$$

Analogous to variational theory in continuous time and space, the calculus of variations in discrete time and continuous space was studied since the 1960s. A *fundamental*

functional optimization problem in discrete time and continuous state is defined as follows:

$$\min_{y} J[y] = \sum_{t=0}^{N} F(t, y(t), y(t+1)), \quad \text{where } y(t) \in R^u. \quad (4)$$

Such studies led to the development of discrete-time Euler-Lagrange equations (Cadzow 1970), the Noether Theorem (Logan 1973), and extensions to integrable systems (Veselov 1988).

Although these results do not consider constraints, side constraints, including Lagrange constraints and isoperimetric constraints (Cadzow 1970), can be included. Constraints are typically handled by using Lagrange multipliers to transform a constrained problem into an unconstrained one, and by applying the Euler-Lagrange condition on the Lagrangian function to get the Euler-Lagrange equations. The conditions obtained are only necessary but not sufficient, as the theory of Lagrange multipliers only yields necessary conditions for continuous optimization.

Unlike (2) and (4), constraints in general control problems (Cadzow 1970) are introduced explicitly by a *control vector* and a *control function* governing the system. The major principle for such control problems is the Pontryagin minimum principle (Dreyfus 1965; Cadzow 1970) that can be derived by first treating the control function as a side constraint, transforming a general problem into a constrained fundamental problem, constructing a Lagrangian function using Lagrange multipliers, and finally solving the Euler-Lagrange equations.

The theory in continuous space applies to continuous and differentiable functions and cannot be used to solve (1) in

discrete space when functions are not continuous and differentiable. To solve (1), we first apply our theory of Lagrange multipliers for discrete constrained optimization (Wah & Wu 1999) to (1) without general constraints and show that a bundle satisfying the discrete-neighborhood saddle-point condition is necessary and sufficient for it to be a local-minimum bundle in its discrete neighborhood. We then establish counterpart discrete-neighborhood Euler-Lagrange equations for (1) and show that the saddle-point condition on a bundle can be partitioned into multiple saddle-point conditions, one for each stage. Due to space limitations, we do not show the extension of our results to the Pontryagin minimum principle for discrete-space general control problems.

Lagrangian theory on discrete constrained optimization

Consider a discrete equality-constrained nonlinear programming problem (NLP):

$$\min_x \quad f(x) \quad \text{where } x \text{ is a vector of} \quad (5)$$
$$\text{subject to} \quad h(x) = 0, \quad \text{discrete variables.}$$

Here, $f(x)$ and $h(x) = (h_1(x), \cdots h_m(x))^T$ are either analytic or procedural. We do not include inequality constraint $g(x) \leq 0$ in (5), as it can be handled easily by transforming the constraint function into an equivalent equality constraint using a a non-negative continuous transformation function H defined as follows:

$$H(g(x)) \begin{cases} = 0 & \text{iff } g(x) = 0, \\ \geq 0 & \text{otherwise.} \end{cases} \quad (6)$$

Function H is easy to design; examples of which include $H(g(x)) = (|g_1(x)|, \cdots, |g_k(x)|)^T$ and $H(g(x)) = (\max(g_1(x), 0), \cdots, \max(g_k(x), 0))^T$. As a result, $g(x) \leq 0$ can be transformed into $H(g(x)) = 0$. Such transformations are not used in conventional Lagrange-multiplier methods in continuous space because the transformed functions are not differentiable at $g(x) = 0$. However, they do not pose any problem in our algorithms because we do not require their differentiability. Moreover, practical search algorithms employing greedy search do not enumerate all possible neighborhood points.

To characterize solutions sought in discrete space, we define $\mathcal{N}(x)$, the *neighborhood* (Aarts & Korst 1989) of point x in discrete space $\mathcal{X}$, as a *finite* user-defined set of points $\{x' \in \mathcal{X}\}$ in such a way that x' is reachable from x in one step, that $x' \in \mathcal{N}(x) \iff x \in \mathcal{N}(x')$, and that it is possible to reach every other x'' starting from any x in one or more steps through neighboring points.

Point $x \in \mathcal{X}$ is a *discrete-neighborhood constrained local minimum* (CLM_{dn}) if it satisfies two conditions: a) x is a feasible point, implying that x satisfies $h(x) = 0$; and b) $f(x) \leq f(x')$ for all feasible $x' \in \mathcal{N}(x)$. Point $x \in \mathcal{X}$ is a *discrete-neighborhood constrained global minimum* (CGM_{dn}) if x is feasible and $f(x) \leq f(x')$ for all $x' \in \mathcal{X}$.

A *generalized Lagrangian function* (Wah & Wu 1999) of (5) is defined as follows:

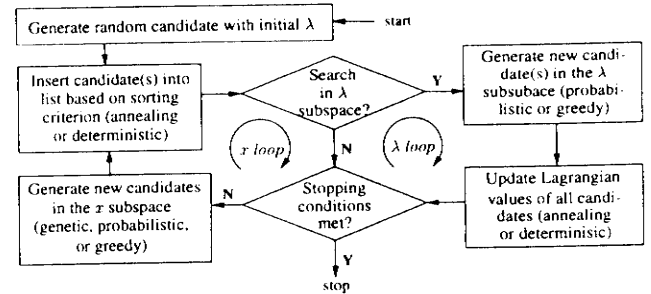$$L_d(x, \lambda) = f(x) + \lambda^T H(h(x)), \quad (7)$$

Figure 5: An iterative procedure to look for SP_{dn} (Wah & Chen 2001).

where $\lambda = (\lambda_1, \cdots, \lambda_m)^T$ is a vector of continuous Lagrange multipliers, and H is defined in (6).

A *direction of maximum potential drop (DMPD)* defined in a discrete neighborhood of $L_d(x, \lambda)$ at point x for fixed λ is a vector that points from x to $x' \in \{x\} \cup \mathcal{N}(x)$ with the minimum L_d:

$$\Delta_x L_d(x, \lambda) = x' - x = (x'_1 - x_1, \cdots, x'_n - x_n)^T$$
$$\text{where } x' = \underset{y \in \mathcal{N}(x) \cup \{x\}}{\operatorname{argmin}} L_d(y, \lambda).$$

A *discrete-neighborhood saddle point* $SP_{dn}(x^*, \lambda^*)$ satisfies the following property similar to that of SP_{cn} (saddle point in continuous neighborhoods):

$$L_d(x^*, \lambda) \leq L_d(x^*, \lambda^*) \leq L_d(x, \lambda^*), \quad (8)$$

for all $x \in \mathcal{N}(x^*)$ and $\lambda \in R^m$. However, SP_{dn} and SP_{cn} are different because they are defined using different neighborhoods.

Theorem 2. *First-order necessary and sufficient conditions for* CLM_{dn} (Wah & Wu 1999). A point in the discrete search space of (5) is a CLM_{dn} iff it satisfies either:

a) the saddle-point condition (8) for any $\lambda \geq \lambda^*$; or

b) the following discrete-space first-order conditions:

$$\Delta_x L_d(x, \lambda) = 0 \text{ and } h(x) = 0. \quad (9)$$

Theorem 2, when applied to solve (5), is stronger than its counterpart continuous-space first-order necessary conditions, which require CLM_{cn} (CLM in continuous neighborhoods) to be regular points and all functions to be differentiable. In contrast, these conditions are not required for CLM_{dn}. Further, the first-order conditions for continuous problems are only necessary and must be augmented by second-order sufficient conditions, whereas the conditions in Theorem 2 are necessary as well as sufficient.

Based on the conditions in Theorem 2, Figure 5 depicts a general procedure to look for SP_{dn} (Wah & Chen 2001). It consists of two loops: the x loop that updates the x variables in order to perform descents of L_d in its x subspace, and the λ loop that updates the λ variables of unsatisfied constraints in order to perform ascents in its λ subspace. The point where the procedure converges can be proved to be a local minimum in the discrete neighborhood of L_d and a local maximum in the continuous neighborhood of λ.

Various implementations have been developed based on the procedure, including the *discrete Lagrangian method*

(DLM) (Wah & Wu 1999) and *constrained simulated annealing* (CSA) (Wah & Wang 1999). DLM entails greedy searches in the x and λ subspaces until all constraints are satisfied, using deterministic rules in accepting candidates. In contrast, CSA generates new probes randomly in one of the x and λ variables, accepts them based on the Metropolis probability if L_d increases along the x dimension and decreases along the λ dimension, and stops updating λ when all the constraints are satisfied.

An important issue in using CSA is the selection of a cooling schedule. (Chen 2001) proves that there exists an optimal cooling schedule that can minimize the expected search time of CSA in finding a solution. An optimal version of CSA with iterative deepening has been developed to find the optimal cooling schedule dynamically (Chen 2001).

Calculus of Variations in Discrete Space

Basic definitions

Solutions to (1) cannot be characterized in ways similar to those of continuous-space problems with differentiable functions. The latter entails strong and weak relative minima that are defined with respect to neighborhoods of open spheres with radius approaching zero asymptotically. Such concept does not exist in problems with discrete variables.

To characterize solutions in discrete space, we define new concepts on neighborhoods and constrained solutions.

Definition 1. $\mathcal{N}_v(s)$, the *discrete neighborhood* (Aarts & Korst 1989) of state vector $s \in \mathcal{Y}$, is a finite user-defined set of states $\{s' \in \mathcal{Y}\}$ such that $s' \in \mathcal{N}_v(s) \iff s \in \mathcal{N}_v(s')$. Further, for any $s^1, s^k \in \mathcal{Y}$, it is possible to find a finite sequence of state vectors $s^1, \cdots, s^k \in \mathcal{Y}$ such that $s^{i+1} \in \mathcal{N}_v(s^i), i = 1, \cdots k-1$.

Definition 2. $\mathcal{N}_b^{(t)}(y)$, the tth-*stage discrete neighborhood of bundle* y for given $\mathcal{N}_v(s)$ and all $t = 0, 1, \cdots, N+1$, is defined as follows:

$$\mathcal{N}_b^{(t)}(y) = \left\{ z \mid z(t) \in \mathcal{N}_v(y(t)) \text{ and } z(i \mid i \neq t) = y(i) \right\}. \quad (10)$$

Intuitively, $\mathcal{N}_b^{(t)}(y)$ includes all bundles that are identical to y in all stages except t, where the state vector is perturbed to a neighboring state in $\mathcal{N}_v(y(t))$. Based on $\mathcal{N}_b^{(t)}(y)$, we define the discrete neighborhood of bundle y to be the union of discrete neighborhoods across each of the $N+2$ stages.

Definition 3. $\mathcal{N}_b(y)$, the *discrete neighborhood of bundle* y, is defined as follows:

$$\mathcal{N}_b(y) = \bigcup_{t=0}^{N+1} \mathcal{N}_b^{(t)}(y). \quad (11)$$

Definition 4. Bundle y is a *discrete-neighborhood constrained local minimum* (CLM_{dn}) of (1) if a) y is feasible, implying that y satisfies all the constraints in (1); and b) $J[y] \leq J[z]$ for all feasible bundles $z \in \mathcal{N}_b(y)$.

Definition 5. A *generalized discrete Lagrangian function* of (1) is:

$$L_d(y, \lambda, \gamma, \mu) = J[y] + \sum_{t=0}^{N+1} \left\{ \gamma^T(t) H(E(t, y(t))) \right. \quad (12)$$

$$\left. + \lambda^T(t) H(G(t, y(t), y(t+1))) \right\} + \mu^T H(I(y)),$$

where $\lambda(t) = (\lambda_1(t), \cdots, \lambda_p(t))^T$, $\gamma(t) = (\gamma_1(t), \cdots, \gamma_r(t))^T$, and $\mu = (\mu_1, \cdots, \mu_q)^T$ are vectors of Lagrange multipliers, and H is a transformation defined in (6).

Definition 6. A *discrete-neighborhood saddle point* $SP_{dn}(y^*, \lambda^*, \gamma^*, \mu^*)$ of (1) is a point that satisfies the following properties for all $y \in \mathcal{N}_b(y^*), \lambda \in R^{(N+2)u}, \gamma \in R^{(N+2)r}$, and $\mu \in R^q$:

$$L_d(y^*, \lambda, \gamma^*, \mu^*) \leq L_d(y, \lambda^*, \gamma^*, \mu^*), \quad (13)$$
$$L_d(y^*, \lambda, \gamma^*, \mu^*) \leq L_d(y^*, \lambda^*, \gamma^*, \mu^*), \quad (14)$$
$$L_d(y^*, \lambda^*, \gamma, \mu^*) \leq L_d(y^*, \lambda^*, \gamma^*, \mu^*), \quad (15)$$
$$\text{and} \quad L_d(y^*, \lambda^*, \gamma^*, \mu) \leq L_d(y^*, \lambda^*, \gamma^*, \mu^*). \quad (16)$$

These equations state that $SP_{dn}(y^*, \lambda^*, \gamma^*, \mu^*)$ is at a local minimum of $L_d(y, \lambda, \gamma, \mu)$ with respect to y and at a local maximum with respect to λ, γ, and μ.

The following result can be derived easily from Definition 6 and Theorem 2.

Lemma 1. Bundle y in the discrete search space of (1) is a CLM_{dn} iff it satisfies:

a) the discrete-neighborhood saddle-point conditions (13)-(16) for any $\lambda \geq \lambda^*$, $\gamma \geq \gamma^*$, and $\mu \geq \mu^*$; or

b) the following discrete-space first-order conditions:

$$\Delta_y L_d(y, \lambda, \gamma, \mu) = 0, \quad (17)$$
$$E(t, y(t)) = 0, \qquad t = 0, \cdots, N+1,$$
$$G(t, y(t), y(t+1)) = 0,$$
$$\text{and} \quad I(y) = 0.$$

Intuitively, Lemma 1 restates the necessary and sufficient conditions in Theorem 2 when (1) is solved as a nonlinear equality-constrained NLP. Although they define the conditions under which the objective value of a feasible bundle is at a local minimum value, they do not exploit node dominance in order to reduce search complexity. In the next section, we introduce node-dominance relations by partitioning the conditions in Lemma 1 in order to reduce the base of the exponential search complexity (last column in Table 1).

Node dominance by distributed necessary and sufficient conditions

Analogous to the Euler-Lagrange conditions in continuous space, we present next the partitioning of the discrete-neighborhood saddle-point condition in (13)-(16) into multiple saddle-point conditions, one for each stage. The partitioned conditions are essentially node-dominance relations which help prune points that are not discrete-neighborhood saddle points in each stage.

Before the main theorem is presented, we define the following to characterize the distributed properties of the solution space of (1).

Definition 7. The tth-stage, $t = 0, \cdots, N + 1$, *distributed discrete Lagrangian function* of (12) is defined as:

$$\Gamma_d(t, y, \lambda, \gamma, \mu) = F(t - 1, y(t - 1), y(t))$$
$$+ F(t, y(t), y(t + 1)) + \gamma(t)^T H(E(t, y(t)))$$
$$+ \lambda(t - 1)^T H(G(t - 1, y(t - 1), y(t)))$$
$$+ \lambda(t)^T H(G(t, y(t), y(t + 1))) + \mu^T H(I(y))$$

with boundary conditions:

$$\Gamma_d(0, y, \lambda, \gamma, \mu) = F(0, y(0), y(1)) + \gamma(0)^T H(E(0, y(0)))$$
$$+ \lambda(0)^T H(G(0, y(0), y(1))) + \mu^T H(I(y))$$
$$\Gamma_d(N + 1, y, \lambda, \gamma, \mu) = F(N, y(N), y(N + 1))$$
$$+ \gamma(N + 1)^T H(E(N + 1, y(N + 1))) + \mu^T H(I(y)).$$

Definition 8. The tth-*stage distributed direction of maximum potential drop* in a discrete neighborhood of y in stage t, $t = 0, \cdots, N + 1$, is a vector that points from y to $y' \in \{y\} \cup \mathcal{N}_b^{(t)}(y)$ with the minimum $\Gamma_d(t, y, \lambda, \gamma, \mu)$:

$$\Delta_{y(t)} \Gamma_d(t, y, \lambda, \gamma, \mu) = y' - y,$$
$$\text{where } y' = \underset{z \in \{y\} \cup \mathcal{N}_b^{(t)}(y)}{\arg\min} \Gamma_d(t, z, \lambda, \gamma, \mu).$$

The main result is summarized as follows.

Theorem 3. *Distributed necessary conditions for* CLM_{dn}. *If bundle* y *is a* CLM_{dn} *in the discrete search space of* (1), *then it satisfies either one of the following sets of conditions for* $t = 0, 1, \cdots, N + 1$:

a) *Discrete-neighborhood Euler-Lagrange equations* (ELE_{dn}):

$$\Delta_{y(t)} \Gamma_d(t, y, \lambda, \gamma, \mu) = 0, \tag{18}$$
$$E(t, y(t)) = 0, \qquad t = 0, \cdots, N + 1 \tag{19}$$
$$\text{and} \quad G(t, y(t), y(t + 1)) = 0. \tag{20}$$

b) *Distributed discrete-neighborhood saddle-point conditions* (DSP_{dn}):

$$\Gamma_d(t, y^*, \lambda^*, \gamma^*, \mu^*) \leq \Gamma_d(t, y', \lambda^*, \gamma^*, \mu^*), \tag{21}$$
$$\Gamma_d(t, y^*, \lambda', \gamma^*, \mu^*) \leq \Gamma_d(t, y^*, \lambda^*, \gamma^*, \mu^*), \tag{22}$$
$$\text{and} \quad \Gamma_d(t, y^*, \lambda^*, \gamma', \mu^*) \leq \Gamma_d(t, y^*, \lambda^*, \gamma^*, \mu^*) \tag{23}$$
$$\text{for any } \gamma' \geq \gamma^* \text{ and } \lambda' \geq \lambda^*.$$

Here, $y' \in \mathcal{N}_b^{(t)}(y^*)$, $\lambda' \in \mathcal{N}_b^{(t)}(\lambda^*)$, $\gamma' \in \mathcal{N}_b^{(t)}(\gamma^*)$, and $\mu' \in R^q$, where y', λ', and γ' represent, respectively, y^*, λ^*, and γ^* perturbed in the tth stage, namely,

$$y' = (y^*(0), \cdots, y^*(t - 1), y'(t), y^*(t + 1), \cdots, y^*(N + 1))$$
$$\lambda' = (\lambda^*(0), \cdots, \lambda^*(t - 1), \lambda'(t), \lambda^*(t + 1), \cdots, \lambda^*(N)),$$
$$\gamma' = (\gamma^*(0), \cdots, \gamma^*(t - 1), \gamma'(t), \gamma^*(t + 1), \cdots, \gamma^*(N + 1)),$$
$$\mathcal{N}_b^{(t)}(\lambda^*) = \{\lambda \mid \lambda(t) \in R^u \text{ and } \lambda(i \mid i \neq t) = \lambda^*(i)\},$$
$$\mathcal{N}_b^{(t)}(\gamma^*) = \{\gamma \mid \gamma(t) \in R^u \text{ and } \gamma(i \mid i \neq t) = \gamma^*(i)\}.$$

The theorem can be proved by examining the functional structure of the conditions in Lemma 1 and by decomposing the conditions in each stage. Due to space limitations, we do not show the proof in this paper.

```
1.   procedure DCV+CSA
2.       initialize starting values of y, λ, γ, μ;
3.       set starting temperature T = T₀ and cooling schedule S;
4.       ρ ← ρ₀ ; /* vector ρ controls the update of μ*/
5.       repeat /* outer loop in iterative scheme */
6.           for t = 0 to N + 1 /* inner loop for SP_dn */
7.               generate a trial point of either y(t), λ(t), or γ(t);
8.               accept the trail point with probability A_T;
9.           end_for
10.          generate a trial point of μ; /* ascents in μ subspace */
11.          accept the trail point with probability A_T;
12.      until stopping condition is satisfied;
13.  end_procedure
```

Figure 6: DCV+CSA: an iterative procedure for finding points that satisfy DSP_{dn} in Theorem 3 using CSA in Figure 5.

There are several salient features of Theorem 3.

a) Unlike classical calculus of variations in continuous space, Theorem 3 does not require the differentiability or continuity of functions.

b) Conditions (18)-(20) in ELE_{dn} are the discrete-space counterpart of the continuous Euler-Lagrange equations, whereas (21)-(23) in DSP_{dn} indicate that finding $N + 2$ discrete-neighborhood saddle points, one in each stage, is necessary for finding a CLM_{dn}. These conditions are the node-dominance relations for pruning states in a search. In particular, (21)-(23) are very easy to implement and are employed in the variational search algorithm presented next.

c) As is discussed in the first section, the use of node dominance helps reduce the search complexity from $O(|\mathcal{Y}|^N)$ to $O(|\mathcal{Y}|N|s|^N)$ (see Table 1). For simplicity, consider $|s|$ to be the number of discrete-neighborhood saddle points in $\mathcal{Y}$ in each stage. Since $|s|$ is much smaller than $|\mathcal{Y}|$, node dominance helps reduce the base of the exponential complexity to a much smaller value. This reduction in complexity is illustrated in Figure 3.

d) In a way similar to that described in the second section, our theory can handle inequality constraints by transforming $g(x) \leq 0$ into an equivalent equality constraint $H(g(x)) = 0$ using transformation H defined in (6).

Discrete-Space Variational Search implementing Node Dominance

Figure 6 outlines DCV+CSA, a heuristic procedure for finding saddle points that satisfy DSP_{dn} in Theorem 3. It uses CSA in Figure 5 with default parameters to search for SP_{dn} in each stage. This is done by looking for local minima in the x subspace of $\Gamma_d(t, y, \lambda, \gamma, \mu)$ in stage t that satisfy (21), and by looking for local maxima in the λ and γ subspaces that satisfy (22) and (23). Once all the stages have been examined, it performs ascents of $L_d(y, \lambda, \gamma, \mu)$ defined in (16) in the μ subspace if there were unsatisfied general constraints. As before, since it is very difficult to determine the optimal cooling schedule for CSA in the inner loop, we apply iterative deepening and double the duration of the cooling schedule S iteratively in order to determine the optimal cooling schedule. The procedure stops when no further improvements can be made in the y, λ, γ and μ subspaces. In general, the inner loop does not have to use CSA and can be any algorithm that looks for local SP_{dn} in a stage.

Experiments on ASPEN

In this section we compare the performance of ASPEN (Automated Scheduling and Planning Environment developed at the Jet Propulsion Laboratory (JPL)) with ASPEN integrated with, respectively, CSA and DCV+CSA. The performance improvements are demonstrated experimentally on a series of spacecraft operation planning application benchmarks.

The ASPEN System

ASPEN (Chien, *et al.* 2000) is an objective-based planning system for automated complex planning and scheduling of spacecraft operations. Such operations involve generating a sequence of low-level parallel spacecraft commands from a set of high-level science and engineering goals.

Using discrete time horizons and a discrete state space, an ASPEN model encodes spacecraft operability constraints, flight rules, spacecraft hardware models, science experiment goals, and operations procedures. It defines various types of *schedule constraints* that may be in procedural form among or within the parallel activities to be scheduled. Such constraints include temporal constraints, decomposition constraints, resource constraints, state dependency constraints, and goal constraints. In addition, the quality of a schedule is defined in a *preference score*, which is a weighted sum of multiple preferences (that may also be in procedural form) to be optimized by the planner. Preferences can be related to the number of conflicts, the number of actions, the value of a resource state, or the value of an activity parameter.

Since ASPEN cannot optimize plan quality and search for feasible plans at the same time, it alternates between its repair-based feasibility planning and the optimization of plan quality.

In the repair phase (Rabideau *et al.* 1999), ASPEN first generates an initial schedule that may not be conflict-free, using an algorithm called forward dispatch. It then searches for a feasible plan from this initial plan, using iterative repairs that try to resolve each conflict individually in the current plan. In each repair iteration, the planner must decide at each *choice point* a conflict to resolve and a conflict-resolution method from a rich collection of repair heuristics.

To improve the quality of plans defined by a preference score, ASPEN uses a preference-driven, incremental, local optimization method. Based on multiple choice points in each iteration, ASPEN provides various optimization heuristics for deciding search directions at choice points.

ASPEN Application Benchmarks

The ASPEN software can be tested on several publicly available benchmarks that schedules parallel spacecraft operations. These benchmarks encode goal-level tasks commanded by science and engineering operations personnel, with a goal of generating high-quality plans as fast as possible. There are four benchmarks tested in this paper:

- The CX1-PREF benchmark (Willis, Rabideau, & Wilklow 1999) models the operations planning of the Citizen Explorer-1 (CX-I) satellite that took data relating to ozone and downlinked its data to ground for scientific analysis. It was used by JPL to explore autonomous command and control software running on the satellite and the ground system, and automated planning and scheduling software (ASPEN) running on the ground system. The benchmark has a problem generator that can generate problem instances of different number of satellite orbits.

- The DCAPS benchmark (Rabideau *et al.* 1997) models the operation of DATA-CHASER shuttle payload, managed by the University of Colorado at Boulder. DATA-CHASER is a science payload primarily used for solar observation, whose payload was scheduled for the STS-85 shuttle flight in August 1997.

- OPTIMIZE and PREF are two benchmarks developed at JPL that come with the licensed release of ASPEN.

Integrating CSA and DCV+CSA in ASPEN

We have tested the performance of DCV+CSA in ASPEN by comparing it to that of ASPEN and ASPEN with CSA.

In our experiments on ASPEN, we allow it to alternate between a repair phase with unlimited number of iterations and an optimization phase with 200 iterations.

In integrating CSA with ASPEN, ASPEN+CSA performs descents of Γ_d by choosing probabilistically among ASPEN's repair and optimizations actions, selecting a random feasible action at each choice point, applying the selected action to the current schedule, and accepting the new schedule based on the Metropolis probability in CSA. In our implementation, we fix the initial temperature to be 1000.0, determine the cooling schedule by iterative deepening (Wah & Chen 2000), set a weight of the objective value in the Lagrangian function by a factor of 100.0 (since the preference score is only between 0 to 1), initialize all Lagrange multipliers to be zero, and increase those multipliers assigned to unsatisfied schedule conflicts in each iteration by 0.1. There are no parameters in determining neighborhoods as in CSA because all probes are generated by ASPEN heuristics.

In integrating DCV with ASPEN+CSA, we need to determine the number of stages used as well as the duration of each stage. In ASPEN, each conflict has an active window bounded by a start time and an end time called the *time points*. Since ASPEN has discrete time horizons, we can either treat each time point as a discrete stage or collapse adjacent time points into a stage.

ASPEN+DCVs+CSA partitions a time horizon statically and evenly into N stages. This simple strategy often leads to an unbalanced number of time points in different stages. During a search, some stages may contain no conflicts to be resolved, whereas others may contain too many conflicts. Such imbalance leads to search spaces of different sizes across different stages and search times that may be dominated by those in a few stages.

To address this issue, ASPEN+DCVd+CSA partitions time points dynamically into stages by adjusting the boundary of stages at run time in order to balance evenly the activities across different stages. This is accomplished by sorting all the time points at the end of the outer loop of DCV+CSA (Line 11 in Figure 6), and by partitioning the time horizon into N stages in such a way that each stage contains approximately the same number (M/N) of time points, where M
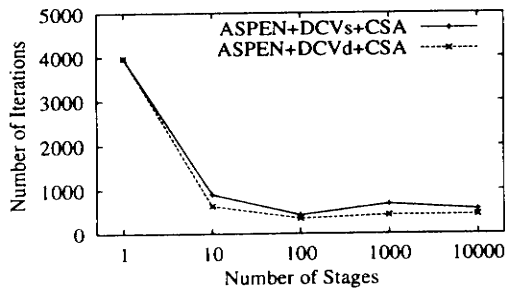
Figure 7: Number of iterations taken by ASPEN+DCVs+CSA and ASPEN+DCVd+CSA to find a feasible plan for an 8-orbit CX1-PREF problem under various numbers of stages.

is the total number of time points in the horizon.

Another important issue to be considered is the number of stages N. Figure 7 shows the number of iterations taken by ASPEN+DCVs+CSA and ASPEN+DCVd+CSA in finding a feasible schedule in solving an 8-orbit CX1-PREF problem. The results show that $N = 100$ is optimal, although the performance is not very sensitive to N when it is large enough. Since other benchmarks lead to similar conclusions, we set $N = 100$ in our experiments.

Experimental Results

In the original implementation, ASPEN commits to every repair/optimize action it generates and never undo any changes of its schedule. In our approach, however, a candidate schedule may not be accepted and the change to the schedule must be undone in that case. Since ASPEN does not support the undo mechanism, we create at each new probe a child process that is a duplicate of the ASPEN program in memory. We then apply the scheduling actions on the child copy, evaluate the candidate schedule, and carry out the same action in the main process only if the new schedule is accepted after evaluation. The CPU overhead of forking a child process is significant, and the CPU time of one iteration in our implementation is about 10 to 20 times longer than that of AS-PEN. However, this CPU overhead will only be marginal with an efficient undo implementation in ASPEN. Therefore, we compare the number of iterations taken, instead of the CPU time, in our experimental results.

Figure 8 compares the performance of ASPEN+DCVd +CSA and ASPEN+DCVs+CSA with respect to that of AS-PEN and ASPEN+CSA on five benchmarks: CX1-PREF with 8 and 16 orbits, DCAPS, OPTIMIZE, and PREF. In each case, we plot the the quality of the best feasible schedule found with respect to the number of search iterations. The results show that ASPEN+DCVd+CSA and AS-PEN+DCVs+CSA are able to find bundles of the same quality one to two orders faster than ASPEN and ASPEN+CSA and much better bundles when they converge. Further, by partitioning time points dynamically into stages, AS-PEN+DCVd+CSA can find better plans in shorter times than ASPEN+DCVs+CSA.
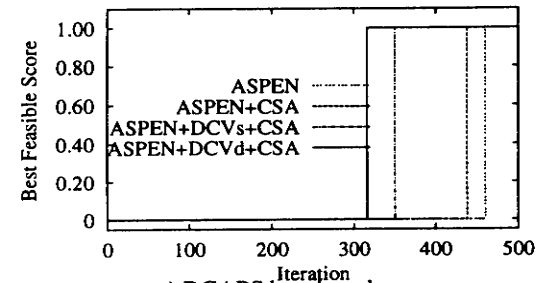
Conclusions

In this paper we have modeled planning problems as dynamic optimization problems and have proposed new node-
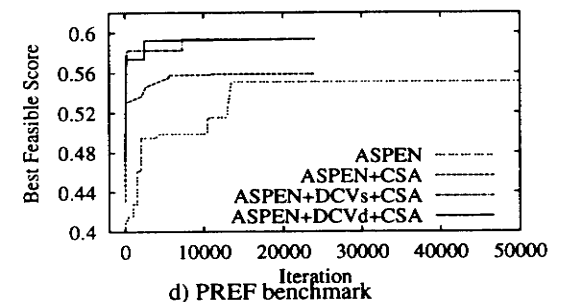

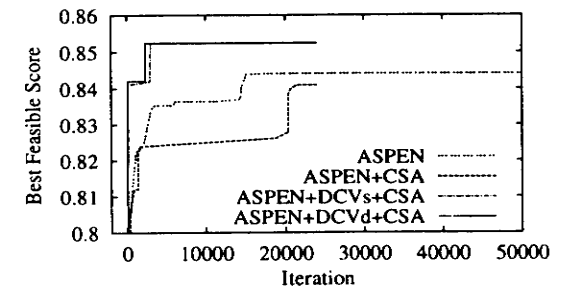a) CX1-PREF benchmark with 8 orbits
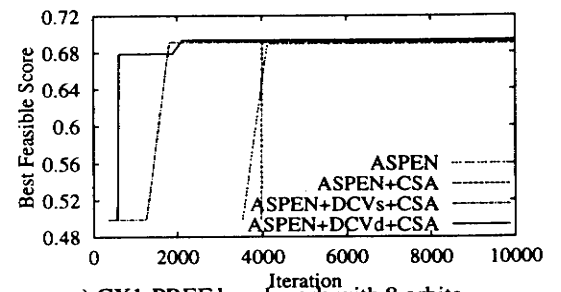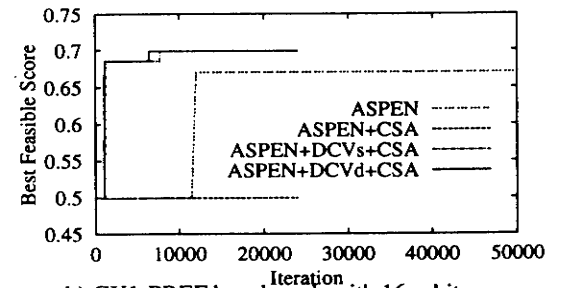
b) CX1-PREF benchmark with 16 orbits

c) DCAPS benchmark

d) PREF benchmark

e) OPTIMIZE benchmark

Figure 8: Quality-time trade-offs in ASPEN, ASPEN+CSA, AS-PEN+DCVs+CSA, and ASPEN+DCVd+CSA on five benchmarks. (All runs involving DCV were terminated at 24,000 iterations.)

dominance relations to improve their solutions. Although path dominance governed by the Principle of Optimality is well studied in dynamic programming, it cannot be applied when there are general constraints that involve variable in multiple stages. On the other hand, node dominance in dynamic optimization is well studied in control theory with continuous and differentiable functions. This exists in the form of calculus of variations when Lagrange multipliers are used to handle nonlinear constraints. However, the use of node dominance in discrete problems with non-differentiable functions is open in the literature.

We have applied the theory of Lagrange multipliers in discrete space in order to allow goal optimization and constraint satisfaction to be handled in a uniform fashion. By exploiting the organization of dynamic optimization problems in stages, we have partitioned the Lagrangian function and have searched for distributed discrete-neighborhood saddle points, one in each stage. Node dominance in each stage is, therefore, implemented by the dominance of saddle points over non-saddle points. Such node dominance lowers the base of the exponential complexity in looking for feasible bundles. Our results on integrating our proposed technique in ASPEN have demonstrated improvements in search complexity as well as plan quality.

Our proposed node dominance can be extended easily to continuous-time planning problems in Figure 1. Similar to that in ASPEN, each conflict in a continuous-time problem is governed by a duration bounded by a continuous start time and an end time. Since the number of conflicts are finite, the number of start and end times is also finite. Hence, the original planning problem can be partitioned into a finite number of stages in such a way that each can be solved as a discrete or a mixed-state constrained optimization problem using Lagrange multipliers.

Similarly, our proposed node dominance can be extended to mixed-state planning problems in Figure 1. This is possible because, once a mixed-state planning problem is partitioned into stages, each stage is a mixed-state constrained optimization problem that can be formulated using Lagrange multipliers. In case of mixed-state functions that are differentiable with respect to continuous variables, we can develop necessary conditions that combine the first-order necessary and sufficient conditions in Theorem 2 and the first-order necessary conditions in continuous-space Lagrange-multiplier theory. In case of functions that are not differentiable with respect to continuous variables, weaker necessary conditions can be developed after the continuous variables have been discretized. We will present results on these extensions in the future.

References

Aarts, E., and Korst, J. 1989. *Simulated Annealing and Boltzmann Machines*. J. Wiley and Sons.

Bellman, R. E. 1957. *Dynamic Programming*. Princeton, NJ: Princeton Univ. Press.

Cadzow, J. A. 1970. Discrete calculus of variations. *Int. J. Control* 11:393–407.

Chen, Y. X., and Wah, B. W. 2002. Calculus of variations in discrete space for constrained nonlinear dynamic optimization. In *Proc. Int'l Conf. on Tools with Artificial Intelligence*, 67–74.

Chen, Y. X. 2001. *Optimal Anytime Search for Constrained Nonlinear Programming*. Urbana, IL: M.Sc. Thesis, Dept. of Computer Science, Univ. of Illinois.

Chien, et al., S. 2000. ASPEN - Automating space mission operations using automated planning and scheduling. In *Proc. SpaceOps*. Toulouse, France.

Dreyfus, S. E. 1965. *Dynamic Programming and the Calculus of Variations*. New York, NY: Academic Press.

Logan, J. D. 1973. First intergal in the discrete calculus of variations. *Aequationes Mathematicae* 9:210–220.

Rabideau, G.; Chien, S.; T. Mann, C. E.; Willis, J.; Siewert, S.; and Stone, P. 1997. Interactive, repair-based planning and scheduling for shuttle payload operations. *Proc. IEEE Aerospace Conf.* 325–341.

Rabideau, G.; Knight, R.; Chien, S.; Fukunaga, A.; and Govindjee, A. 1999. Iterative repair planning for spacecraft operations in the ASPEN system. *Proc. Int'l Symp. on Artificial Intelligence Robotics and Automation in Space*.

Veselov, A. P. 1988. Integrable discrete-time systems and difference operators. *Funct. Anal. Appl.* 22:83–93.

Wah, B. W., and Chen, Y. X. 2000. Optimal anytime constrained simulated annealing for constrained global optimization. *Sixth Int'l Conf. on Principles and Practice of Constraint Programming* 425–439.

Wah, B. W., and Chen, Y. X. 2001. Chapter 10: Genetic algorithms for nonlinear constrained optimization. In Yao, X., and Sarker, R., eds., *Evolutionary Optimization*. Kluwer Academic Pub. 253–275.

Wah, B. W., and Wang, T. 1999. Simulated annealing with asymptotic convergence for nonlinear constrained global optimization. *Principles and Practice of Constraint Programming* 461–475.

Wah, B. W., and Wu, Z. 1999. The theory of discrete Lagrange multipliers for nonlinear discrete optimization. *Principles and Practice of Constraint Programming* 28–42.

Willis, J.; Rabideau, G.; and Wilklow, C. 1999. The Citizen Explorer scheduling system. *Proc. of the IEEE Aerospace Conf.*