Discrete-Space Lagrangian Optimization for Multi-Objective Temporal Planning

Content areas: planning, mathematical foundations, constraint programming

Abstract

In this paper, we study multi-objective temporal planning problems in discrete time and space formulated as single-objective dynamic optimization problems with a minimax objective function. We propose efficient node-dominance relations for pruning states that will not lead to locally optimal plans. Based on the theory of Lagrange multipliers in discrete space, we present the necessary and sufficient conditions for locally optimal plans, partition the Lagrangian function into distributed Lagrangian functions, one for each stage, and show the distributed necessary conditions in the form of local saddle-point conditions in each stage. 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, and that can converge asymptotically to Pareto optimal plans. 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.

1 Introduction

Many planning and scheduling applications can be formulated as nonlinear constrained *dynamic optimization problems* with variables that evolve over time. In this paper we focus on multi-objective AI planning problems that can be formulated using a discrete planning horizon, discrete state vectors representing positive and negative facts, and constraints representing preconditions and effects of actions.

An important property commonly considered necessary for any feasible candidate solution to a multi-objective optimization problem is *Pareto optimality* [Steuer, 1986]. A *Pareto optimal set* consists of *Pareto optimal solutions* (POS) that are not dominated by any other solutions, where solution y *dominates* solution x if x is worse than or equal to y in all objectives, with at least one strictly worse. Most search algorithms look for POS in the Pareto optimal set.

There are several approaches for finding POS in unconstrained space. Consider a problem of optimizing $F(x)$ consisting of a vector of k objective functions:

$$\min_x \quad F(x) = (f_1(x), f_2(x), \cdots, f_k(x))^T. \quad (1)$$

The easiest and widely used approach is the weighted-sum method [Steuer, 1986] that combines the multiple objectives linearly into a single objective using a vectors of weights, one for each objective. A new POS can be found by varying the weights and by solving the single-objective problem for each combination of weights. The main disadvantage of this method is that all POS in the Pareto optimal set can only be generated when all the objective functions are convex. In the special case of looking for local POS (with respect to other local POS in the neighborhood), the convexity assumption is satisfied when the objective functions are continuous and differentiable, but fails in general for discrete objective functions considered in this paper. In the latter, there may not exist weights for some local POS with respect to other local POS in their discrete neighborhoods.

The norm method is based on minimizing the relative distance from a candidate solution to an ideal reference solution vector $(f_1^*, \cdots, f_k^*)$. It transforms the multiple objectives into the following single objective with integer p:

$$\min_x \left[\sum_{i=1}^{k} w_i \left(\frac{f_i(x) - f_i^*}{f_i^*} \right)^p \right]^{\frac{1}{p}}, \quad (2)$$

where each POS is associated with a fixed combination of weights. It represents a family of methods because different distance measures are obtained by varying p, and more POS are expected to be found in nonconvex problems using larger p. However, for finite p, it cannot guarantee that all POS be found, even for all possible combinations of weights.

The minimax method [Steuer, 1986] can potentially generate all POS for nonconvex problems by minimizing the maximum of the weighted criteria in the feasible set, leading to a scalar objective at point x as follows:

$$\min_x \left\{ \max_{i=1}^{k} [w_i f_i(x)] \right\}. \quad (3)$$

This is a special case of (2) when $p = \infty$ and $f_i^* = 0$. In contrast to norm methods using finite p, only the minimax approach guarantees that all POS be reachable.

Due to its generality, we use the minimax approach in this paper to handle multiple objectives. A multi-objective nonlinear constrained planning problem in discrete space and time

can be formulated as a *single-objective dynamic optimization problem* with equality constraints as follows:

$$\min_{y} \quad J[y] = \max_{i=1}^{k}[w_i J_i(y)] \tag{4}$$

$$\text{such that} \quad E(t, y(t)) = 0, \qquad t = 0, \cdots, N+1$$
$$G(t, y(t), y(t+1)) = 0,$$
$$\text{and} \quad I[y] = 0,$$

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 t; $y(t) = (y_1(t), \cdots, y_u(t))^T$ is a u-element *state vector* in discrete space $\mathcal{Y}$; $(J_1[y], J_2[y], \cdots, J_k[y])^T$ is a k-element vector of objective functions; $E = (E_1, \cdots, E_r)^T$ is a r-component vector of *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*.

A local constraint in (4) involves only local state variables in one stage; a Lagrange constraint involves state variables in adjacent stages; and a general constraint involves state variables across more than two stages. Note that constraints may involve conditions on individual states, preconditions on an action, conditions to be maintained throughout an action, and post-conditions to be achieved by an action, and that J, E, G and I are *not* necessarily continuous and differentiable.

A solution $y = (y(0), y(1), \cdots, y(N+1))$ to (4) consists of u discrete-stage curves (also called sequences), one for each dynamic state variable. Following conventional terminologies in continuous control theory, we call y a *bundle* (or a vector of curves), and $J_i[y]$, the ith *functional* defined as a mapping from y to a value in $\mathcal{R}$.

Existing methods for solving discrete-state discrete-time planning problems can be classified into three categories: a) Systematic search methods that explore the entire state space are expensive, as they are enumerative in nature. Examples include Graphplan, STAN, and PropPLAN. b) Heuristically guided 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. c) Transformation methods transform a 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.

There are two major issues in existing work that we plan to address in this paper. First, with the exception of AS-PEN [Chien, *et al.*, 2000], all existing planners use either a single objective or no objective at all in their formulations, whereas ASPEN allows multiple objectives in a weighted sum. As a weighted sum cannot lead to all POS in the Pareto optimal set unless all its objectives are convex, we propose to use a more general objective that minimizes the maximum of the weighted criteria in the feasible set. The use of a minimax objective in constrained optimization has not been done before due to its non-differentiability. Second, all existing planners search in the original problem space that grows exponentially with respect to the number of variables, or apply heuristics to prune the search space that may not lead to feasible plans. To address this issue, we propose new dominance relations that prune infeasible space and help reduce the base of the exponential complexity without sacrificing feasibility. We further apply constrained simulated annealing (CSA) [Wah & Wang, 1999], which can converge asymptotically to a constrained global optimum, in order to look for POS in the reduced space.

2 Dominance Relations

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 and node dominance.

A special case of (4) with local and Lagrange constraints but without general constraints can be solved by dynamic programming. The *Principle of Optimality* 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. It leads to *path dominance* that allows the optimal path between s and c to dominate other suboptimal paths. Path dominance leads to search complexity that is polynomial in the number of states in the problem.

When general constraints $I[y] = 0$ are present in (4), 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-dominating partial bundle may not. Without path dominance, 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 in the number of states. [1]

Another type of dominance that does not involve the Principle of Optimality is a node-dominance relation between two states c_1 and c_2. State c_2 is said to dominate c_1 ($c_2 \rightarrow c_1$) when c_1 can be pruned safely because it cannot lead to better feasible objective values than those of c_2. Node dominance in stage t is different from path dominance because it only depends on state variables in stage t and not on paths (that depend on states in earlier stages) leading to these states. It helps reduce the overall number of bundles enumerated because any feasible bundle should only involve dominating but not dominated states in each stage. Note that node dominance is only necessary for global optimality in (4).

Figure 1 illustrates the effect of pruning in each stage due to node dominance. As an illustration, consider an enumerative algorithm for finding an optimal bundle in a simple $(N+2)$-stage problem, with an initial state in the first stage, a final state in the last stage, and a discrete search space $\mathcal{Y}$ in stage $t = 1, \cdots, N$. Without node dominance, the algorithm will enumerate all possible combinations of bundles of $|\mathcal{Y}|$ states in stage t, leading to a worst-case complexity of $O\left(|\mathcal{Y}|^N\right)$. In contrast, by partitioning the search into stages and by applying node dominance in each stage, the search

---

[1] The implicit assumption is that (4) 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 $O\left(|\mathcal{Y}|^N\right)$, where $|\mathcal{Y}|$ is the number of states in each stage.
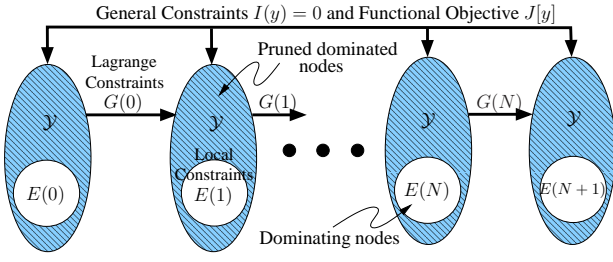
Figure 1: The pruning of dominated states in each stage leads to a smaller search space in finding optimal bundles across different stages. The dominating states are shown to satisfy the local constraints in the figure, and stronger dominance can be achieved by using the distributed saddle-point conditions shown in Section 3.

can be restricted to only dominating states in each stage. Assuming that node dominance leads to $|\mathbf{s}|$ dominating states in each stage and that it takes a worst-case complexity of $|\mathcal{Y}|$ in stage t to find a dominating state in $\mathbf{s}$, the worst-case complexity will be $O\left(N|\mathcal{Y}| \times |\mathbf{s}|^N\right)$. Since $|\mathbf{s}|$ is generally much smaller than $|\mathcal{Y}|$, node dominance helps reduce the base of the worst-case exponential complexity to a much smaller value.

Figure 1 illustrates a weak form of node dominance in which dominating states only satisfy the local constraints in each stage. In this paper, we present stronger dominance relations found by formulating (4) in a Lagrangian function with discrete variables, based on the theory of Lagrange multipliers in discrete space [Wah & Wu, 1999], and by partitioning the Lagrangian function into multiple Lagrangian functions, each involving only local variables in a stage. The dominating states are those that satisfy the local saddle-point condition of the partitioned Lagrangian function. This new approach of optimization in discrete space reduces the search of a locally optimal bundle to that of a saddle point in a discrete neighborhood of the bundle, and further to the search of multiple local saddle points, one in each stage.

3 Node Dominance by Distributed Saddle-Point Conditions

In this section we describe our proposed node-dominance relations in the form of distributed necessary conditions for locally optimal bundles.

To handle inequality constraints, we first define a nonnegative continuous transformation function H that transforms inequality constraint $g(x) \leq 0$ into an equivalent equality constraint $H(g(x)) = 0$, where:

$$H(y) \begin{cases} = 0 & \text{iff } y = 0, \\ \geq 0 & \text{otherwise.} \end{cases} \quad (5)$$

Function H is easy to design; examples of which include $H(g(x)) = [|g_1(x)|, \cdots, |g_k(x)|]^T$ and $H(g(x)) = [\max(g_1(x), 0), \cdots, \max(g_k(x), 0)]^T$. 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 here because we do not require their differentiability. Moreover, practical algorithms employing greedy search do not enumerate all possible neighborhood points.

User-defined neighborhoods. To characterize constrained solutions of (4), we first define $\mathcal{N}_b^{(t)}(y)$, the neighborhood in a stage, and the discrete neighborhood of a bundle. Intuitively, $\mathcal{N}_b^{(t)}(y)$ includes all bundles that are identical to y in all stages except t, where the state vector in stage t is perturbed to a neighboring state in $\mathcal{N}_v(y(t))$. The discrete neighborhood of bundle y is the union of discrete neighborhoods across each of the $N + 2$ stages.

Definition 1. $\mathcal{N}_v(s)$, the *discrete neighborhood* 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 $s^1, \cdots, s^k \in \mathcal{Y}$ 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:

$$\mathcal{N}_b^{(t)}(y) = \left\{ z \mid z(t) \in \mathcal{N}_v(y(t)) \text{ and } z(i) = y(i); i \neq t \right\}. \quad (6)$$

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

$$\mathcal{N}_b(y) = \bigcup_{t=0}^{N+1} \mathcal{N}_b^{(t)}(y). \quad (7)$$

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

Necessary and sufficient conditions. The following two definitions define the terms needed in the conditions.

Definition 5. Using H defined in (5), a *generalized discrete Lagrangian function* of (4) is:

$$L_d(y, \gamma, \lambda, \mu) = J[y] + \sum_{t=0}^{N+1} \left\{ \gamma^T(t) H\left(E(t, y(t))\right) \right. \quad (8)$$

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

where $\gamma(t) = (\gamma_1(t), \cdots, \gamma_r(t))^T \in \mathcal{R}^r$, $\lambda(t) = (\lambda_1(t), \cdots, \lambda_p(t))^T \in \mathcal{R}^p$, and $\mu = (\mu_1, \cdots, \mu_q)^T \in \mathcal{R}^q$ are vectors of Lagrange multipliers.

Definition 6. A *discrete-neighborhood saddle point* $SP_{dn}(y^*, \gamma^*, \lambda^*, \mu^*)$ of (4) is a point that satisfies the following property for all $y \in \mathcal{N}_b(y^*); \gamma, \gamma^* \in \mathcal{R}^{(N+2)r}; \lambda, \lambda^* \in \mathcal{R}^{(N+2)p}$; and $\mu, \mu^* \in \mathcal{R}^q$:

$$\begin{aligned} L_d(y^*, \gamma^*, \lambda^*, \mu^*) &\leq L_d(y, \gamma^*, \lambda^*, \mu^*), \\ L_d(y^*, \gamma, \lambda^*, \mu^*) &\leq L_d(y^*, \gamma^*, \lambda^*, \mu^*), \quad (9) \\ L_d(y^*, \gamma^*, \lambda, \mu^*) &\leq L_d(y^*, \gamma^*, \lambda^*, \mu^*), \\ L_d(y^*, \gamma^*, \lambda^*, \mu) &\leq L_d(y^*, \gamma^*, \lambda^*, \mu^*). \end{aligned}$$

These inequalities state that $(y^*, \gamma^*, \lambda^*, \mu^*)$ is at a local minimum of $L_d(y, \gamma, \lambda, \mu)$ with respect to y and at a local maximum with respect to γ, λ and μ. The following necessary and sufficient condition was first derived in [Shang & Wah, 1998] and shows the one-to-one correspondence between SP_{dn} and CLM_{dn}.

Lemma 1. Bundle y in the discrete search space of (4) is a CLM_{dn} iff it satisfies the discrete-neighborhood saddle-point conditions in (9).

Partitioned Lagrangian functions. The Lagrangian function in (8) can be partitioned into multiple sub-functions, one for each stage. The sub-functions are useful for characterizing the properties of the distributed solution space of (4).

Definition 7. The tth-stage distributed discrete Lagrangian function of (8), $t = 1, \cdots, N$, is:

$$\begin{aligned}
\Gamma_d(t, y, \gamma, \lambda, \mu) &= J[y] + \gamma(t)H(E(t, y(t))) &(10)\\
&+ \lambda(t-1)H(G(t-1, y(t-1), y(t)))\\
&+ \lambda(t)H(G(t, y(t), y(t+1))) + \mu H(I[y])
\end{aligned}$$

with boundary functions:

$$\begin{aligned}
\Gamma_d(0, y, \gamma, \lambda, \mu) &= J[y] + \gamma(0)H(E(0, y(0)))\\
&+ \lambda(0)H(G(0, y(0), y(1))) + \mu H(I[y]),\\
\Gamma_d(N+1, y, \gamma, \lambda, \mu) &= \gamma(N+1)H(E(N+1, y(N+1)))\\
+J[y] + \mu H(I[y]) &+ \lambda(N)H(G(N, y(N), y(N+1))).
\end{aligned}$$

Distributed necessary conditions. By applying Lemma 1 on the distributed Lagrangian function in (10), we obtain necessary conditions on locally optimal bundles with respect to bundles in their neighborhoods defined in (7). These conditions help prune dominated states in each stage that do not satisfy the conditions (cross-shaded area in Figure 1). They are necessary but not sufficient because states that satisfy these conditions may not satisfy the general constraints.

Theorem 1. *Distributed necessary conditions for CLM_{dn}.* If y is a CLM_{dn} in the discrete space of (4), then it satisfies the following distributed discrete-neighborhood saddle-point conditions for $t = 0, 1, \cdots, N+1$:

$$\begin{aligned}
\Gamma_d(t, y^*, \gamma^*, \lambda^*, \mu^*) &\leq \Gamma_d(t, y', \gamma^*, \lambda^*, \mu^*);\\
\Gamma_d(t, y^*, \gamma', \lambda^*, \mu^*) &\leq \Gamma_d(t, y^*, \gamma^*, \lambda^*, \mu^*); &(11)\\
\Gamma_d(t, y^*, \gamma^*, \lambda', \mu^*) &\leq \Gamma_d(t, y^*, \gamma^*, \lambda^*, \mu^*);
\end{aligned}$$

where

$$\begin{aligned}
\gamma' &= (\gamma^*(0), \cdots, \gamma^*(t-1), \gamma'(t), \gamma^*(t+1), \cdots, \gamma^*(N+1));\\
\lambda' &= (\lambda^*(0), \cdots, \lambda^*(t-1), \lambda'(t), \lambda^*(t+1), \cdots, \lambda^*(N+1));\\
y' &= (y^*(0), \cdots, y^*(t-1), y'(t), y^*(t+1), \cdots, y^*(N+1)).\\
\mathcal{N}_b^{(t)}(\gamma^*) &= \{\gamma \mid \gamma(t) \in \mathcal{R}^r \text{ and } \gamma(i \mid i \neq t) = \gamma^*(i)\};\\
\mathcal{N}_b^{(t)}(\lambda^*) &= \{\lambda \mid \lambda(t) \in \mathcal{R}^p \text{ and } \lambda(i \mid i \neq t) = \lambda^*(i)\};
\end{aligned}$$

$\gamma', \gamma^* \in \mathcal{R}^{(N+2)r}$ and $\lambda', \lambda^* \in \mathcal{R}^{(N+2)p}$ for all $y' \in \mathcal{N}_b^{(t)}(y^*)$; $\gamma' \in \mathcal{N}_b^{(t)}(\gamma^*)$ and $\lambda' \in \mathcal{N}_b^{(t)}(\lambda^*)$ are perturbed in the tth stage.

The theorem is a straightforward application of the conditions in Lemma 1 on the partitioned Lagrangian function [Chen & Wah, 2003]. The conditions are stronger than node dominance based on local constraints alone because they also require dominating states in each stage to satisfy the Lagrange constraints (third condition in (11)).

As the number of combinations of dominating states across different stages is substantially smaller than the number of

```
1.  procedure DCV+CSA
2.      set starting values of y, λ, γ, μ;
3.      set starting temperature T = T₀ and cooling schedule S;
4.      set N_T; /* number of probes per stage per T */
5.      repeat /* outer loop in iterative scheme */
6.          for t = 0 to N + 1 do /* find SP_dn for stage t*/
7.              for i = 0 to N_T do
8.                  generate a trial point of y(t), λ(t), or γ(t);
9.                  accept the trail point with probability A_T;
10.             end_for
11.         end_for
12.         generate a trial point of μ; /* ascents in μ subspace */
13.         accept the trail point with probability A_T;
14.         reduce temperature using a geometric schedule;
15.     until stopping condition is satisfied;
16. end_procedure
```

Figure 2: DCV+CSA: an iterative procedure for finding points that satisfy (11) in Theorem 1 using CSA.

combinations of possible states across different stages, Theorem 1 leads to significant reduction in search complexity.

Our approach is similar to that of *calculus of variations* in continuous control theory [Cadzow, 1970] that partitions a Lagrangian function of a multi-stage problem in continuous space into multiple sub-functions and that develops first-order necessary (Euler-Lagrange) conditions in each stage that govern local optimality. Our approach, however, works in discrete space and does not require the differentiability and continuity of functions. Note that, although we have presented the theory based on discrete-neighborhood saddle points, the theory can be expressed in terms of first-order descent directions, similar to that of classical calculus of variations in continuous space based on first-order gradient directions.

4 Asymptotically Convergent Algorithm

Theorem 1 defines the conditions for CLM_{dn} that are local optimal bundles with respect to their neighborhoods. Such solutions can be found by performing greedy descents of $\Gamma_d(t, y, \lambda, \gamma, \mu)$ in the $y(t)$ subspace in stage t, and by performing greedy ascents in the $\lambda(t)$ and $\gamma(t)$ subspaces. Once all the stages have been examined, greedy ascents of $L_d(y, \lambda, \gamma, \mu)$ are performed in the μ subspace (last condition in (9)) if there were unsatisfied general constraints.

To find POS that are *constrained global minima* (CGM_{dn}) to (4), we employ *constrained simulated annealing* (CSA) [Wah & Wang, 1999] that generates new probes randomly in each stage, and accepts them based on the Metropolis probability A_T if Γ_d (and L_d as well) increases along the y dimension and decreases along the λ and γ dimensions. The search then generates new probes in the μ dimension after iterating over all stages, and accepts them based on the Metropolis probability if L_d increases. The search stops updating λ, γ, and μ when all the constraints are satisfied. Note that, since Γ_d includes both the objective and the constraints, the algorithm looks for local POS that optimizes the objective as well as satisfying the constraints at the same time.

The acceptance probability A_T of new trial points is controlled by the decreasing temperature T. It can be proved [Wah & Wang, 1999] that, by using a logarithmic cooling schedule and by modeling the search as a finite in-

homogeneous Markov chain, the search converges asymptotically to CGM_{dn} of (4).

Figure 2 outlines DCV+CSA, an algorithm that performs CSA one stage at a time before performing ascents in the μ subspace. We use a geometric cooling scheduling $T = \alpha \times T$, where $0 < \alpha < 1$ in Line 14, since the logarithmic cooling schedule is too slow in practice.

As it is very difficult to determine the optimal cooling rate α in the cooling schedule, we try to select the duration of the cooling schedule iteratively. Based on an observed monotonically non-decreasing relationship between the success probabilities of obtaining a solution and the average completion times of CSA, we apply iterative deepening that schedules multiple runs of CSA, using a set of geometrically increasing durations, in order to minimize the expected completion time (to a constant factor) [Wah & Chen, 2000]. When the search stops, one has a high probability that a POS has been found. Note that it is not possible to guarantee that a POS be found in finite time, as CSA only converges to a POS asymptotically.

5 Pareto Optimal Solutions in ASPEN

In this section we show the performance of integrating DCV+CSA in ASPEN (Automated Scheduling and Planning Environment [Chien, *et al.*, 2000] developed at the Jet Propulsion Laboratory (JPL)) in order to find POS. ASPEN is an objective-based planner 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 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 interleaves its repair-based feasibility planning with the optimization of plan quality. In the repair phase, It 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.

We have compared performance using three publicly available benchmarks on scheduling parallel spacecraft operations: OPTIMIZE, PREF and CX1-PREF. 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. OPTIMIZE (10 objectives) and PREF (50 objectives) are two benchmarks developed at JPL that come with the licensed release of ASPEN. The CX1-PREF benchmark [Willis, Rabideau, & Wilklow, 1999] (7 objectives) models the operations planning of the Citizen Explorer-I (CX-I) satellite that took data relating to ozone and downlinked its data to ground for scientific analysis. It has a problem generator that can generate problem instances of different number of satellite orbits.

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

Next, we tested ASPEN+CSA, a version of ASPEN that uses CSA to choose probabilistically among ASPEN's repair and optimizations actions, select a random feasible action at each choice point, apply the selected action to the current schedule, and accept the new schedule based on the Metropolis probability in CSA. In our implementation, we fix the initial temperature to be 1000, determine the cooling schedule by iterative deepening [Wah & Chen, 2000], initialize all Lagrange multipliers to zero, and increase those multipliers of 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.

Last, we tested ASPEN+DCV+CSA, a version of ASPEN that uses DCV+CSA in Figure 2 to look for discrete-neighborhood saddle points in each stage that satisfy (11). Since the performance of the algorithm depends on the number of stages, we study two ways of collapsing adjacent time points in the discrete time horizon of ASPEN into stages.

In ASPEN+DCVs+CSA, we partition the 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 15 in Figure 2), 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 is the total number of time points in the horizon.

Figure 3 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 the best performance is achieved when $N = 100$. Since other benchmarks also lead to similar conclusions, we set $N = 100$ in our experiments.

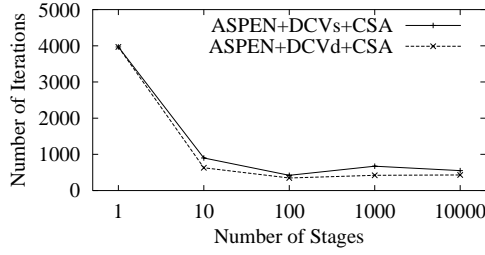Figure 4 compares the performance of ASPEN with respect

Figure 3: 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 different number of stages.

a) CX1-PREF benchmark with 8 orbits

b) CX1-PREF benchmark with 16 orbits
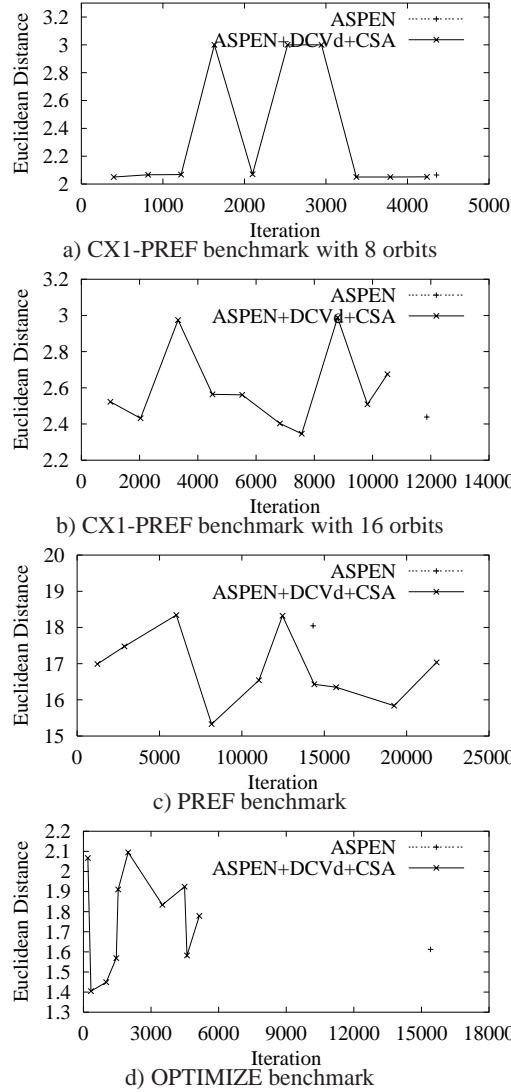

c) PREF benchmark

d) OPTIMIZE benchmark

Figure 4: POS found by ASPEN+DCVd+CSA as compared to the solution found by ASPEN formulated using a weighted sum. Little overhead is incurred by DCV+CSA, and iterations of both methods are comparable.

to that of ASPEN+DCVd+CSA. In ASPEN+DCVd+CSA, we plot ten different POS found using random sets of weights (between 0 and 100) in the minimax formulation. In each case, we show the Euclidean distance between the objective vectors of the schedule found to the Utopian objective vec-

Table 1: Weighted-sum solution of ASPEN and 10 POS found by ASPEN+DCVd+CSA on a CX1-PREF problem with 8 orbits.

J	ASPEN Sol.	ASPEN+DCVd+CSA									
		S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
$J1$	1	1	1	1	1	1	1	1	1	1	1
$J2$	0	0	0	0	0	0	0	0	0	0	0
$J3$	0.9910	0.9911	0.9915	0.9916	0.9951	0.9924	0.9998	0.9961	0.9911	0.9911	0.9913
$J4$	3.15e-06	0	0	0	5.49e-05	4.49e-06	0	1.31e-06	0	0	0
$J5$	0.745	0.776	0.742	0.740	0	0.735	0	0	0.776	0.776	0.773
$J6$	1	1	1	1	1	1	1	1	1	1	1
$J7$	1	1	1	1	1	1	1	1	1	1	1

tor in which all objective functions are of the maximum value 1.0. The results show that ASPEN+DCVd+CSA can find a POS one to two orders faster than ASPEN and can generate multiple POS.

Table 1 further illustrates the various seven-objective solutions found on the 8-orbit CX1-PREF problem. It can be verified that, due to the existence of S3 and S10, there exists no combination of weights that make S2 a global minimum in the weighted-sum objective used by ASPEN. This is true because, in order for S2 to be better than S10 on the weighted sum, the weight on J3 must be at least 155 times larger than that on J5; however, this will make S2 worse than S3 in terms of the weighted sum. As a result, it is not possible for ASPEN to find S2 using an objective based on a weighted sum.

References

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

[Chen & Wah, 2003] Chen, Y. X., and Wah, B. W. 2003. Automated planning and scheduling using calculus of variations in discrete space. In *Proc. Int'l Conf. on Automated Planning and Scheduling*.

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

[Shang & Wah, 1998] Shang, Y., and Wah, B. W. 1998. A discrete Lagrangian based global search method for solving satisfiability problems. *J. of Global Optimization* 12(1):61–99.

[Steuer, 1986] Steuer, R. 1986. *Multiple criteria optimization*. New York: John Wiley and Sons, Inc.

[Wah & Chen, 2000] 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 & Wang, 1999] 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 & Wu, 1999] 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, Rabideau, & Wilklow, 1999] Willis, J.; Rabideau, G.; and Wilklow, C. 1999. The Citizen Explorer scheduling system. *Proc. of the IEEE Aerospace Conf.*