Distributed Constraint Problem Solving and Reasoning in Multi-Agent Systems
W. Zhang and V. Sorge (Eds.)
IOS Press, 2004

The Evaluation of Partitioned Temporal Planning Problems in Discrete Space and its Application in ASPEN

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

Abstract. In this paper, we study the partitioning and evaluation of discrete-space temporal planning problems and illustrate our techniques on ASPEN, an objective-based planner developed at the Jet Propulsion Laboratory for the automated planning and scheduling of complex spacecraft control operations. We formulate these planning problems as single or multi-objective dynamic optimization problems and propose the necessary and sufficient extended saddle point condition (ESPC) that governs the correctness of locally optimal plans. We then decompose the ESPC into partitioned ESPC, one for each stage, and show that they are necessary and sufficient collectively. By utilizing these partitioned conditions, we present efficient search algorithms whose complexity, despite exponential, has a much smaller base as compared to that without using the conditions. 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 single- and multi-objective temporal 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. We study the partitioning of these problems into subproblems related by global constraints, and develop methods for resolving the global constraints after the subproblems have been solved.

Consider *ASPEN* [4], an objective-based planning system for the automated planning and scheduling of complex spacecraft operations. It involves generating a sequence of parallel low-level spacecraft control commands from a set of high-level science and engineering goals [8]. Using a discrete time horizon 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

```
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";;
prefer linearly less resource power total value;; //optimization objective
// initial plan
A1 act_1 start_time = 0; duration = 15;; A1 act_1 start_time = 20; duration = 10;;
A2 act_2 start_time = 30; duration = 10;; A2 act_2 start_time = 50; duration = 10;;
```

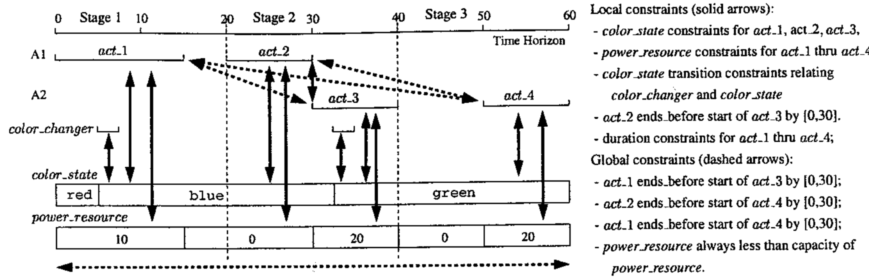


Figure 1: A toy example from ASPEN [4] in which the goal is to find a valid plan that completes activities *act_1*, *act_2*, *act_3*, and *act_4* while minimizing the total power usage. By partitioning the horizon into three stages [0,20], [21,40], [41,60], 192 constraints are localized, leaving only 6 global constraints that span across multiple stages. In this toy example, the number of iterations is reduced from 16 taken by ASPEN to 12 after the problem is partitioned and the subproblems solved by ASPEN.

may be in procedural form among or within the parallel activities to be scheduled. Such constraints include temporal, decomposition, resource, state-dependency, and goal constraints. In addition, the quality of a plan is defined in a preference score, which is a weighted sum of multiple preferences (that may also be procedural) 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, and the value of an activity parameter.

Figure 1 illustrates a toy example planning problem from ASPEN. It involves scheduling four activities over a discrete horizon of 60 seconds in order to satisfy various constraints that relate the activities and *power_resource*, *color_state*, and *color_changer*, while trying to minimize the total *power_resource* used. By indexing variables by time $t = 0, \ldots, 60$, Boolean variable $s_i(t)$ (*resp.* $a_i(t)$ and $e_i(t)$) defines that act_i starts (*resp.* is active and ends) at t. Let $cc(t)$ be the *color_changer* activity at t, which can be set to `red`, `blue`, `green`, or `not_active`; $c(t)$ be the *color_state* that can be set to `red`, `blue`, or `green`; $w(t) = 10(a_1(t) + a_2(t)) + 20(a_3(t) + a_4(t))$ be the *power_usage*; and $p(t)$ be the *power_supply* at t. The following illustrates a small portion of the constraints encoded, where $A \Longrightarrow B$ can be translated into constraint $(1 - A) + B \geq 0$, and the objective is to minimize $\sum_{t=0}^{60} w(t)$:

$$\begin{aligned}
a_1(t) = 1 &\quad\Longrightarrow\quad c(t) = 2; \quad \text{// } \textit{color_state} \text{ constraint for } \textit{act_1} \\
c(t-1) \neq c(t) &\quad\Longrightarrow\quad cc(t-1) = c(t); \quad \text{// } \textit{color_state} \text{ transition constraint} \\
w(t) \;\leq\; p(t) &\quad\leq\quad 25, \;\forall\, t = 0, \ldots, 60; \quad \text{// } \textit{power_resource} \text{ capacity constraint} \\
e_1(t_1) = 1 \wedge s_3(t_2) = 1 &\quad\Longrightarrow\quad 0 \leq t_2 - t_1 \leq 30; \quad \text{// } \textit{act_1} \text{ ends_before start of } \textit{act_3} \text{ by } [0,30].
\end{aligned}$$

Using this formulation, the problem has a total of 198 constraints and one objective function.

Figure 2 illustrates another example that shows the 3,687 constraints of an initial (infeasible) plan generated by ASPEN [4] in solving CX1-PREF with 16 orbits.

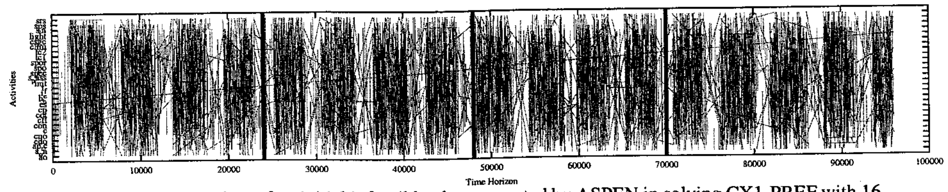


Figure 2: The 3,687 constraints of an initial infeasible plan generated by ASPEN in solving CX1-PREF with 16 orbits. Each constraint is shown as a line that relates two activities (labeled in the y-axis) scheduled at two time instances in the horizon (x-axis). The partitioning of the horizon into four stages (separated by bold vertical lines) leads to 3,580 local constraints and 107 global constraints.

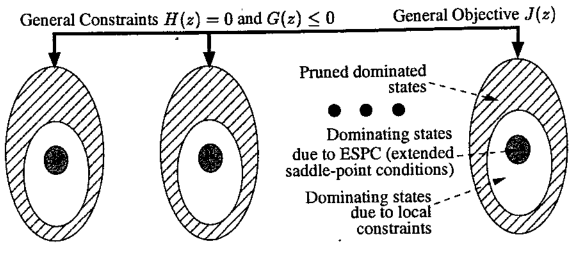


Figure 3: The pruning of states that do not satisfy local constraints and ESPC in each stage leads to a significant reduction in the base of the exponential complexity across all stages.

A general approach for solving multi-stage problems is to apply dynamic programming and to resolve the constraints in a stage-by-stage fashion. Unfortunately, dynamic programming cannot be applied in the problems illustrated in Figures 1 and 2 because a global constraint in these problems may run across any two arbitrary stages and not always between adjacent stages. As a result, a partial feasible plan that dominates another partial feasible plan in one stage will fail to hold when the dominating plan violates a global constraint in a later stage. Moreover, even if the stages can be partitioned in such a way that allows global constraints to only relate states in adjacent stages, the number of states to be enumerated in each stage may be so large that makes it impossible to apply the Principle of Optimality.

Our approach in this paper for solving planning problems is illustrated in Figures 1 and 2. It divides a planning problem into stages, evaluates each subproblem independently by resolving their local constraints, and resolves those violated global constraints spanning across multiple stages at the end. It is attractive for solving temporal planning problems because many of their constraints and objectives are related to activities with temporal locality.

In our approach, since the complexity of resolving the global constraints is exponential and depends on the Cartesian product of the solution spaces of all the stages, the base of the exponential complexity can be reduced if one can limit the solution space of each stage beforehand. For instance, the search space of a stage can be reduced by finding dominating nodes that satisfy its local constraints (the first inner ellipse in each stage of Figure 3). In Figure 1, the partitioning of the horizon into three stages leads to 192 local constraints and six global constraints. Similarly, partitioning the horizon into four stages in Figure 2 leads to 3,580 local constraints and 107 global constraints.

In addition to reducing the problem complexity, a major benefit of partitioning is that it leads to smaller but similar planning subproblems. As a result, existing planners can be employed to solve these subproblems with little or no modification. Without the need to develop new planners to solve each subproblem, pruning techniques in existing planners, such as constraint propagation and schemes for handling exceptions in execution, can be employed. Further, new planners developed in the future can be integrated easily in our approach.

Existing methods for solving discrete problems do not exploit the partitioning approach because there is no effective method except expensive trial and error for resolving violated global constraints after the partitioned subproblems have been solved. For a similar reason, existing planners do not exploit partitioning in their searches. These planners rely on global information and do not have mechanisms to combine the solutions of partitioned subproblems into global solutions.

Existing planners based on systematic searches explore the entire state space and are not amenable to partitioning because their locally feasible or optimal plans in a partitioned space may not satisfy all the global constraints. These include UCPOP [15], an early goal-directed planner based on the Partial Order Causal Link (POCL) technique; Graphplan [1] that searches a planning graph in order to minimize the length of a parallel plan; STAN [13], an efficient implementation of Graphplan; PropPLAN [5], a planner based on a naive breadth-first search of ordered binary decision diagrams; and System R [12], a systematic search method based on regression that solves one goal at a time.

Existing planners based on local searches employ heuristic guidance functions to search in discrete space. They do not work well on partitioned plans because their guidance heuristics are computed over the entire horizon in order to estimate the distance from a state to the goal state. Examples include HSP [2], a hill-climbing search using heuristic values obtained by solving a relaxed problem; FF [6], an enforced hill-climbing search using heuristic values obtained by solving a relaxed Graphplan problem; AltAlt [14], a hybrid planner on top of STAN and HSP; GRT [18] (and its extension to MO-GRT [19]), a two-phase planner that first estimates the distances between domain facts and goals, before searching by a simple best-first strategy; and ASPEN [4], a repair-based local-search method that can handle discrete temporal and metric constraints and that optimizes multiple objectives in a weighted sum.

Last, planners based on transformations convert a problem into a constrained optimization or satisfaction problem before solving it by existing solvers. They are not amenable to partitioning because they rely on solvers that do not support partitioning. Examples include SATPLAN [9] that transforms a planning problem into a satisfiability (SAT) problem, Blackbox [10] that transforms a planning graph [1] into a SAT problem, and ILP-PLAN [11] that transforms a planning problem into an integer linear programming (ILP) problem with discrete metric constraints and optimization objectives.

In this paper, we formulate in Section 2 a planning problem as a discrete constrained optimization problem. Although this is not a customary representation for planning problems, it is needed in developing a formal mathematical foundation for resolving global constraints. We present in Section 3 our theory of extended saddle-point condition (ESPC) in discrete space and its decomposition into partitioned conditions. We then describe our implementation of the partitioned conditions in Section 4 and their application on improving ASPEN in Section 5. Finally, conclusions are drawn in Section 6.

2 Mathematical Formulation of Discrete Constrained Optimization

Our formulation assumes that the discrete horizon is partitioned in $N+1$ stages, with u_t local variables, m_t local equality constraints, and r_t local inequality constraints in stage t, $t = 0, \ldots, N$. Such partitioning decomposes the discrete variable vector $y \in \mathcal{D}^w$ of the problem into $N+1$ subvectors $y(0), \ldots, y(N)$, where $y(t) = (y_1(t), \ldots, y_{u_t}(t))^T$ is a u_t-element *state vector* in discrete space at stage t, and $y_i(t)$ is the i^{th} *dynamic state variable* in stage t. A solution to such a problem is a *plan* that consists of the assignments of all variables in y. A single-objective formulation of the problem is as follows:

$$\begin{array}{llllll}(P_t): & \min\limits_y & J(y) & & & \\ & \text{subject to} & h^{(t)}(y(t)) = 0, & g^{(t)}(y(t)) \le 0, & t = 0, \ldots, N, & \text{(local constraints)}, \\ & \text{and} & H(y) = 0, & G(y) \le 0, & & \text{(global constraints)}.\end{array} \tag{1}$$

Here, $h^{(t)} = \left(h_1^{(t)}, \ldots, h_{m_t}^{(t)}\right)^T$ and $g^{(t)} = \left(g_1^{(t)}, \ldots, g_{r_t}^{(t)}\right)^T$ are local-constraint functions that involve $y(t)$ and time 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 and time in two or more 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 the functions are not required to be continuous and differentiable.

A planning problem may also involve the optimization of one or more objectives. An important property commonly considered necessary for any feasible candidate solution to a multi-objective optimization problem is *Pareto optimality* [20]. A *Pareto optimal set* consists of *Pareto optimal solutions* (POS) that are not dominated by any other solutions, where solution y *dominates* solution y' if y' is worse than or equal to y in all objectives, with at least one strictly worse. Most search algorithms look for one or more POS in the Pareto optimal set.

There are several approaches for finding POS in unconstrained space. Consider a problem of optimizing $J(y)$ that consists of a vector of k objective functions:

$$\min_y \quad J(y) = (J_1(y), J_2(y), \cdots, J_k(y))^T. \tag{2}$$

The first class of methods are those that transform the multi-objective functions into a single objective and try to find only one POS. The easiest and widely used approach is the weighted-sum method [20] 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 $(J_1^*, \cdots, J_k^*)$. It transforms the multiple objectives into the following single objective with integer p:

$$\min_y \quad J(y) = \left[\sum_{i=1}^{k} w_i \left(\frac{J_i(y) - J_i^*}{J_i^*}\right)^p\right]^{\frac{1}{p}}, \tag{3}$$

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 a larger p. However, for finite p, the method cannot guarantee that all POS be found, even for all possible combinations of weights, unless its objectives are convex.

The minimax method [20, 7] 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 y as follows:

$$\min_y \quad J(y) = \max_{i=1}^{k} w_i J_i(y). \tag{4}$$

This is a special case of (3) in which $p = \infty$ and $J_i^* = 0$. In contrast to norm methods using finite p, only the minimax approach guarantees that all POS be reachable. We adopt this approach in (1) because our criterion for selecting a suitable objective in an optimization is that a systematic search can lead to all POS in the Pareto optimal set.

3 Extended Saddle-Point Conditions in Discrete Space

Consider the following discrete optimization problem P_d:

$$\begin{aligned} (P_d): \quad & \min_y \quad f(y), \quad y \in \mathcal{D}^w \\ & \text{subject to} \quad h(y) = 0 \text{ and } g(y) \le 0. \end{aligned} \tag{5}$$

The goal of solving P_d is to find a constrained local minimum y with respect to $\mathcal{N}_d(y)$, the discrete neighborhood of y.

Definition 1. A user-defined *discrete neighborhood* $\mathcal{N}_d(y)$ of $y \in \mathcal{D}^w$, is a finite user-defined set of states $\{y' \in \mathcal{D}^w\}$ such 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.

Definition 2. Point y^* is a *constrained local minimum in the discrete neighborhood* (CLM_d) of P_d if y^* is feasible and $f(y^*) \le f(y)$ for all feasible $y \in \mathcal{N}_d(y^*)$.

There are two distinct features of CLM_d. First, the set of CLM_d of a problem is neighborhood dependent because it depends on the user-defined discrete neighborhood; that is, y may be CLM_d with respect to $\mathcal{N}_d(y)$ but may not be with respect to $\mathcal{N}'_d(y)$. Although the choice of neighborhoods does not affect the validity of a search as long as a consistent definition is used throughout, it may affect the time to find a CLM_d. Second, a discrete neighborhood has a *finite* number of points. Hence, the verification of y to be CLM_d with respect to $\mathcal{N}_d(y)$ can be done by comparing its objective value against that of its *finite* number of discrete neighboring points. This feature allows the search of a descent direction in discrete neighborhood to be done by enumeration or greedy search, rather than by differentiation.

Before we state the main theorem, we define a new Lagrangian function in discrete space:

Definition 3. The ℓ_1-penalty function of P_d in (5) is defined as follows:

$$L_d(y, \alpha, \beta) = f(y) + \alpha^T |h(y)| + \beta^T \max(0, g(y)), \tag{6}$$

where α and β are the *extended Lagrange multipliers*.

Theorem 1. *Necessary and sufficient extended saddle point condition (ESPC) on CLM_d of P_d.* [21] Suppose $y^* \in \mathcal{D}^w$ is a point in the discrete space of P_d. Then y^* is a CLM_d of P_d iff there exist finite $\alpha^* \ge 0$ and $\beta^* \ge 0$ such that, for any $\alpha^{**} > \alpha^*$ and $\beta^{**} > \beta^*$, the following saddle-point condition is satisfied:

$$L_d(y^*, \alpha, \beta) \le L_d(y^*, \alpha^{**}, \beta^{**}) \le L_d(y, \alpha^{**}, \beta^{**}) \tag{7}$$

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

The proof of the theorem can be found in the references [23, 21].

The inequalities in (7) state that $(y^*, \alpha^{**}, \beta^{**})$ is at a local minimum of $L_d(y, \alpha, \beta)$ with respect to y and at a local maximum with respect to α and β.

Based on Theorem 1, we can now solve P_t in (1) by partitioning it into subproblems. We first show that plan y^*, a CLM_d with respect to its discrete neighborhood $\mathcal{N}_d(y^*)$, satisfies the ESPC in Theorem 1. To solve (1) efficiently, we define a separable 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 Lagrange multipliers.

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

Definition 4. Given $\mathcal{N}_d(y(t))$, the discrete neighborhood of $y(t)$ in stage t, we define $\mathcal{N}_p(y)$, the *separable discrete neighborhood of plan* y, as follows:

$$\mathcal{N}_p(y) = \bigcup_{t=0}^{N} \mathcal{N}_p^{(t)}(y) = \bigcup_{t=0}^{N} \left\{ y' \;\middle|\; y'(t) \in \mathcal{N}_d(y(t)) \text{ and } y'(i \mid i \ne t) = y(i) \right\}, \tag{8}$$

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

Definition 5. The ℓ_1-penalty function of P_t in (1) is defined as follows:

$$\begin{aligned} L_d(y, \alpha, \beta, \gamma, \eta) = {} & J(y) + \sum_{t=0}^{N} \left\{ \alpha(t)^T |h^{(t)}(y(t))| + \beta(t)^T \max(0, g^{(t)}(y(t))) \right\} \\ & + \gamma^T |H(y)| + \eta^T \max(0, G(y)), \end{aligned} \tag{9}$$

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 Lagrange multipliers.

Next, we show that (7) can be partitioned into a set of necessary and sufficient conditions.

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

$$\begin{aligned} \Gamma_d^{(t)}(y^*, \alpha(t), \beta(t), \gamma^{**}, \eta^{**}) & \le \Gamma_d^{(t)}(y^*, \alpha(t)^{**}, \beta(t)^{**}, \gamma^{**}, \eta^{**}) \\ & \le \Gamma_d^{(t)}(y, \alpha(t)^{**}, \beta(t)^{**}, \gamma^{**}, \eta^{**}), \end{aligned} \tag{10}$$

$$L_d(y^*, \alpha^{**}, \beta^{**}, \gamma, \eta) \le L_d(y^*, \alpha^{**}, \beta^{**}, \gamma^{**}, \eta^{**}), \tag{11}$$

for all $y \in \mathcal{N}_p^{(t)}(y^*)$ 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

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

Proof. We prove the theorem by showing the equivalence of ESPC in (7) and those in (10) and (11).

"$\Rightarrow$" part: Given y^* that satisfies (7), we show that it also satisfies (10) and (11). Since for all $t = 0, \ldots, N$, any $y \in \mathcal{N}_p^{(t)}(y^*)$ is also a point in $\mathcal{N}_p(y^*)$; hence, the inequality on the right of (10) is implied by the inequality on the right of (7). The inequality on the left of (10) and that in (11) are obvious, as all the constraints are satisfied at y^*.

"$\Leftarrow$" part: We prove this part by contradiction. Assuming that y^* satisfies (10) and (11) but not (7), the inequality on the left of (7) cannot be violated because the inequalities on the left of (10) and (11) imply that all local and global constraints are satisfied. Therefore, it must be the inequality on the right of (7) that is not satisfied at y^*. That is, there exist $y \in \mathcal{N}_p(y^*)$ and a unique t' where $y \in \mathcal{N}_p^{(t')}(y^*)$ (according to the definition of $\mathcal{N}_p(y)$ in (8)) such that:

$$L_d(y^*, \alpha^{**}, \beta^{**}, \gamma^{**}, \eta^{**}) \not\leq L_d(y, \alpha^{**}, \beta^{**}, \gamma^{**}, \eta^{**}). \tag{12}$$

This implies that the inequality on the right of (10) is not satisfied at $t = t'$, which contradicts our assumption that y^* satisfies (10) and (11). Note that t' exists because the neighborhood in (8) is separable. These two parts prove the correctness of the theorem. □

By using a separable 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 stages. By reducing the solution space in each subproblem through the search of extended saddle points, Theorem 2 leads to a significant reduction in the base of the exponential complexity in finding CLM_d.

4 Global Search Implementing ESPC in Discrete Space

An important aspect of Theorem 1 is that it suffices to find any $\alpha^{**} > \alpha^*$ and $\beta^{**} > \beta^*$ in order to satisfy the ESPC. Such a property allows the solution of P_d to be found iteratively by looking for a local minimum y^* of $L_d(y, \alpha, \beta)$ with respect to points in $\mathcal{N}_d(y^*)$ in an inner loop, and for any $\alpha^{**} > \alpha^*$ and $\beta^{**} > \beta^*$ in an outer loop.

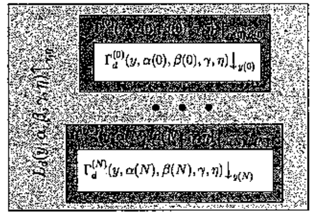
Figure 4a shows the pseudo code that implements the conditions in Theorem 1. The inner loop looks for local minima of $L_d(y, \alpha, \beta)$ in its discrete neighborhoods with respect to y, whereas the outer loop performs ascents on α and β for unsatisfied global constraints and stops when a CLM_d has been found.

The iterative search can be extended to the conditions in Theorem 2. In Figure 4b, the two inner nested loops of stage t look for a local saddle point of $\Gamma_d^{(t)}(y, \alpha(t), \beta(t), \gamma, \eta)$. This is done by updating y, $\alpha(t)$, 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 $y(t)$ that solves the following discrete optimization problem in stage t:

$$\begin{aligned}\min_{y(t)} \quad & J(y) + \gamma^T H(y) + \eta^T G(y) \\ \text{subject to} \quad & h^{(t)}(y(t)) = 0 \quad \text{and} \quad g^{(t)}(y(t)) \leq 0.\end{aligned} \tag{13}$$

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

a) Implementation of Theorem 1



b) Implementation of Theorem 2

Figure 4: Simple iterative implementation of ESPC to look for CLM_d of P_d and that of partitioned ESPC to look for CLM_d of P_t.

Since this is a well-defined optimization problem, any existing solver with little modification can be used to solve it. In the next section, we apply ASPEN to solve partitioned planning subproblems in the form of (13).

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.

A search using the above simple implementations may get stuck in an infeasible region when the objective is too small or when the Lagrange multipliers and/or constraint violations are too large. In this case, increasing the Lagrange multipliers will further deepen the infeasible region, making it impossible for a local-descent algorithm in the inner loops to escape from this region.

To address this issue, we can change either the ascent algorithm in the two outer loops of Figure 4b or its descent algorithm in the innermost loops. The ascent algorithm can be changed to allow increases as well as decreases of Lagrange multipliers α, β, γ, and η. The goal of decreases is to "lower" the barrier in the Lagrangian function in order for local descents in the innermost loops to escape from an infeasible region. Note that α, β, γ, and η should be increased gradually in order to help the search escape from local minima of $L_d(y, \alpha, \beta, \gamma, \eta)$. Once α, β, γ, and η reach their maximum thresholds, they can be scaled down and the search repeated. In our partitioned implementation of ASPEN described in the next section, we have set upper bounds on Lagrange multipliers and scale the multipliers once they reach the upper bounds. However, we have found that periodic decreases of the multipliers before they reach the upper bounds is not necessary because the solution space has many feasible solutions and the search never gets stuck in an infeasible region.

In a similar way, the descent algorithm in the innermost loops can be changed to allow descents as well as ascents. In temporal planning problems, the exact gradient direction of functions may not be available because the functions are not in closed form. As a result, a randomly generated probe does not likely follow descent directions, and a deterministic descent procedure may get stuck easily in infeasible local minima. To cope with this issue, probes generated may be accepted based on stochastic criteria in order to allow descents as well as ascents. In the next section, we describe our partitioned implementation of ASPEN [3] that accepts probes with larger Lagrangian values according to the Metropolis probability.

5 Experimental Results on a Partitioned Implementation of ASPEN

In this section we show experimental results on implementing our partitioned search in Theorem 2 in ASPEN.

ASPEN alternates between a repair phase and an optimization phase because it cannot optimize plan quality and search for feasible plans at the same time. In the repair phase [17], ASPEN generates an initial plan that may have conflicts and searches for a feasible plan from this initial plan, using iterative repairs to resolve conflicts. In a repair step, the planner must decide at each *choice point* a conflict to resolve and a conflict-resolution method from a rich collection of repair heuristics. Next, in the optimization phase, ASPEN uses a preference-driven, incremental, local optimization to optimize plan quality defined by a preference score. It decides the best search direction at each choice point, based on information from multiple choice points. In our experiments, we allow ASPEN to alternate between a repair phase with an unlimited number of iterations and an optimization phase with 200 iterations.

We have compared the performance of the various implementations using OPTIMIZE, PREF, DCAPS, and CX1-PREF. These four publicly available benchmarks on scheduling parallel spacecraft operations 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 [22] (7 objectives) models the operations planning of the Citizen Explorer-1 (CX-1) 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. Last, the DCAPS benchmark [16], managed by the University of Colorado at Boulder, models the operation of DATA-CHASER shuttle payload. Since it has no preferences, we define a preference of one for a feasible solution and zero otherwise.

Figure 5 shows ASPEN+PART(N, PARTITION_STRATEGY), a partitioned implementation of ASPEN for solving planning problems in N stages. Here, we set a weight of 100 in both the single and minimax objective (4) in the Lagrangian function (since the preference score is between 0 to 1), and initialize all Lagrange multipliers to zeroes.

In generating a new plan from the current plan during descents of $\Gamma_d^{(t)}$ (Line 8 in Figure 5), ASPEN chooses probabilistically among its repair and optimization actions, selects a random feasible action at each choice point, and applies the selected actions to the current plan.

As is discussed in the last section, since many of the objectives and constraints in complex spacecraft applications are not differentiable, a new plan generated does not likely follow descent directions, and a local descent of the Lagrangian function may get stuck easily in infeasible local minima. To address this issue, ASPEN+PART(N, PARTITION_STRATEGY) employs the Metropolis probability A_T to determine whether to accept a new plan (Line 9 in Figure 5). Using a parameter called *temperature* T, it accepts the new plan with larger $\Gamma_d^{(t)}$ based on the following A_T, with the acceptance probability decreasing as T decreases:

$$A_T(\mathbf{y}, \mathbf{y}') = exp\left(-\frac{(L_d(\mathbf{y}') - L_d(\mathbf{y}))^+}{T}\right), \tag{14}$$

where $\mathbf{y} = (y, \alpha, \beta, \gamma, \eta)$; $\mathbf{y}' = (y', \alpha, \beta, \gamma, \eta)$; and $(a)^+ = a$ if $a > 0$ and $(a)^+ = 0$ otherwise for all $a \in \mathcal{R}$. We have used a geometric cooling schedule $T_{new} = c \cdot T_{old}$ because the logarithmic cooling schedule is too slow. In our experiments, $c = 0.8$ and the initial temperature is 1,000.


```
/* Partitioned implementation of ASPEN in N stages partitioned using PARTITION_STRATEGY */
1. procedure ASPEN+PART(N, PARTITION_STRATEGY)
2.     generate initial plan y and a partitioning of the horizon into N stages;
3.     set initial temperature T to 1000 and all Lagrange multipliers α, β, γ, and η to zeroes;
4.     repeat
5.         num_descents ← 1;
6.         for t = 1 to N
7.             for k = 1 to num_descents
8.                 generate a new plan y' in a child process;
9.                 evaluate Γ_d^(t) and the Metropolis probability A_T(y, y') controlled by T;
10.                if Γ_d^(t) is accepted
11.                    apply the action in the main process, namely, y ← y';
12.                    update Lagrange multipliers α(t) and β(t) on unsatisfied local constraints;
13.                end_if
14.            end_for
15.        end_for
16.        update Lagrange multipliers γ and η on unsatisfied global constraints;
17.        num_descents ← min(100, num_descents * 2);
18.        reduce temperature T_new ← T_old × c, where c ← 0.8;
19.        if PARTITION_STRATEGY is DYN_P, then dynamically repartition the stages;
20.    until no change in y, α, β, γ, and η in an iteration;
21. end_procedure
```

Figure 5: An iterative procedure for finding points that satisfy Theorem 2 using N stages partitioned using PARTITION_STRATEGY. The Metropolis probability is used to stochastically accept a probe with a worse Lagrangian value during descents (Line 9).

The stochastic approach proposed above requires the modified planner to generate a candidate plan, determine whether to accept it based on the Metropolis probability, and reverse the changes made when the candidate is not accepted. However, the reversal of changes is difficult in ASPEN because ASPEN commits to every repair/optimization action it generates. To address this issue, we create a child process that is a duplicate of the ASPEN program in memory before we evaluate a candidate plan. We apply the scheduling actions on the child copy, evaluate the plan, and carry out the same actions in the main process only if the new plan is accepted. Although the CPU overhead of forking a child process is significant and amounts to 10-20 times longer than the time of a normal iteration, this overhead will be marginal with an efficient implementation of undo in ASPEN. Moreover, since we measure the number of iterations according to the number of plans evaluated, including discarded ones, our results will accurately reflect the overhead in ASPEN if the undo operation were actually implemented.

The Lagrange multipliers are updated in each iteration in Line 12 of Figure 5 by increasing the multipliers of unsatisfied constraints by 0.1. As is discussed in the last section, we set a maximum threshold of 1000 for each Lagrange multiplier and divide all of them by 1000 when one of them reaches the maximum.

Two other important issues that must be addressed in our partitioned implementation are the number of stages used and the duration of each. In ASPEN, a conflict has an active window bounded by a start time and an end time called the *time points*. Adjacent time points can be collapsed into a stage, since ASPEN has discrete time horizons.

We have studied both the static and dynamic partitioning of stages. In static partitioning, ASPEN+PART(N, STATIC$_P$) partitions the horizon statically and evenly into N stages. This simple strategy often leads to an unbalanced number of time points in different stages. During

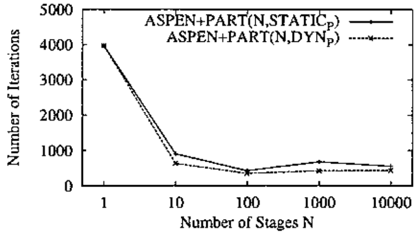


Figure 6: Number of iterations taken by static and dynamic partitioning in ASPEN+PART(N, PARTITION_STRATEGY) to find a feasible plan for the 8-orbit CX1-PREF problem.

a search, some stages may contain no conflicts to be resolved, whereas others may contain too many conflicts. Such an 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 achieve a better balance of activities across stages, ASPEN+PART(N, DYN$_P$) adjusts the boundary of stages dynamically. This is accomplished by finding M, the number of time points in the horizon related to conflicts, at the end of the outer loop (Line 15 in Figure 5) and by partitioning the horizon into N stages in such a way that each stage contains approximately the same number (M/N) of such time points (Line 19 in Figure 5). To determine the best N, Figure 6 plots the number of iterations taken by static and dynamic partitioning in finding a feasible plan of the 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 larger than 100. Since other benchmarks lead to similar conclusions, we set $N = 100$ in the following experiments.

Figure 7 compares the performance of the original ASPEN, ASPEN+PART(1, STATIC$_P$), ASPEN+PART(100, STATIC$_P$), and ASPEN+PART(100, DYN$_P$) on the five benchmarks described earlier, based on a single-objective formulation. In each graph, we plot the quality of the best feasible plan found with respect to the number of search iterations. The results show that our partitioned implementations are able to find plans of the same quality one to two orders faster than ASPEN and ASPEN+PART(1, STATIC$_P$) and much better plans when they converge. Further, dynamic partitioning can find better plans in shorter times than those found by static partitioning.

Figure 8 compares the performance of ASPEN and ASPEN+PART(100, DYN$_P$), where the latter is based on a multi-objective formulation. We plot ten different POS found by ASPEN+PART(100, DYN$_P$), using random sets of weights between 0 and 100 in the minimax formulation (4). In each case, we show the Euclidean distance between the objective vectors of the plan found to the Utopian objective vector in which all objective functions are of the maximum value 1.0. The results show that ASPEN+PART(100, DYN$_P$) 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 makes 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.

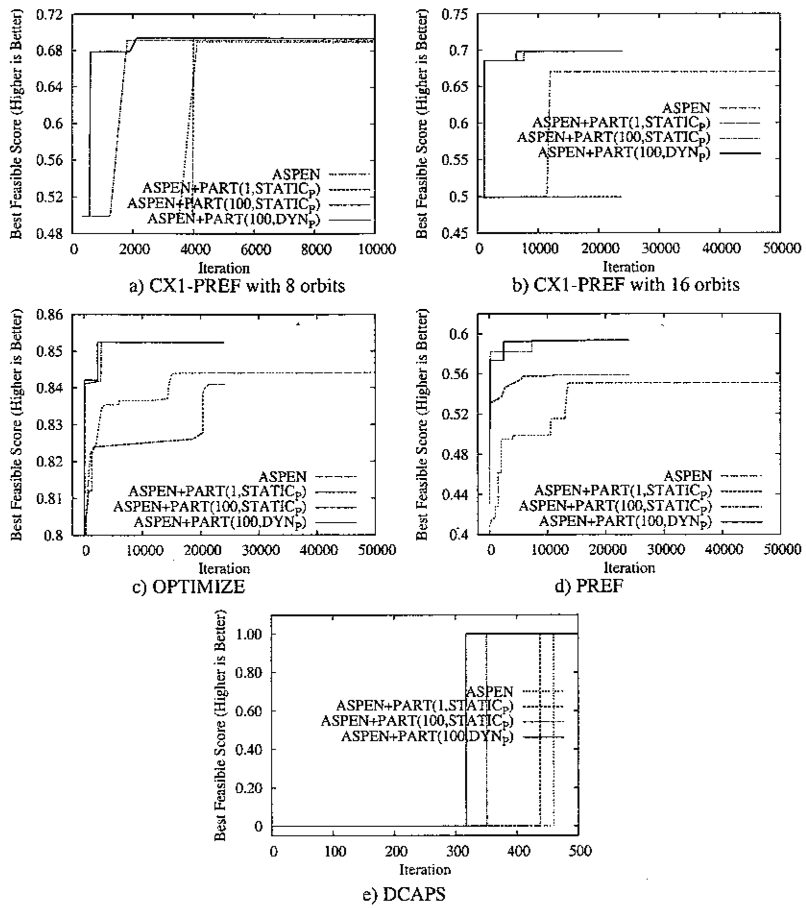


Figure 7: Quality-time comparisons of ASPEN, ASPEN+PART(1, STATIC$_P$), ASPEN+PART(100, STATIC$_P$), and ASPEN+PART(100, DYN$_P$). (All runs involving ASPEN+PART were terminated at 24,000 iterations and used the Metropolis probability to accept probes with worse Lagrangian value during descents. The preference score of DCAPS is one when a feasible solution is found and zero otherwise.)

6 Conclusions

In this paper, we have presented the extended saddle point condition in discrete space and its decomposition into partitioned conditions that collectively are necessary and sufficient for any constrained local minimum in discrete constrained optimization. The theory leads to an efficient iterative scheme for resolving global constraints across partitioned subproblems and for finding partitioned saddle points in each partitioned subproblem. We apply the partitioned conditions on the discrete-space ASPEN planner for solving partitioned planning benchmarks and demonstrate significant improvements, both in terms of the quality of the plans generated and the execution times to find them.

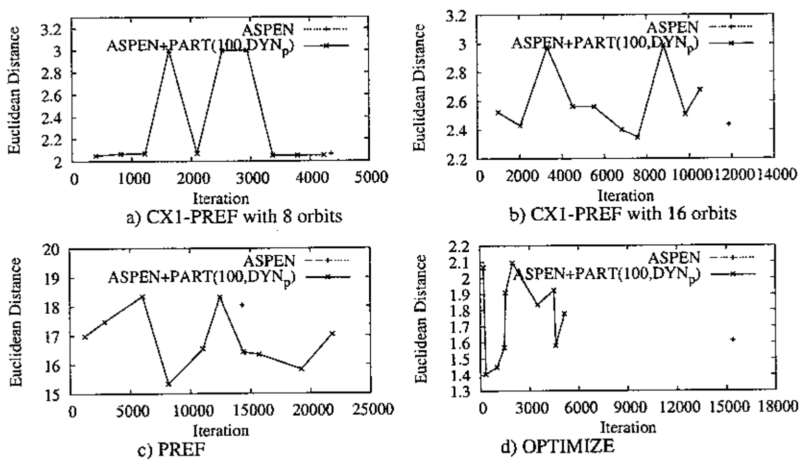


Figure 8: POS found by ASPEN+PART(100, DYN_P) as compared to the solution found by ASPEN formulated using a weighted sum.

Table 1: Weighted-sum solution of ASPEN and 10 POS found by ASPEN+PART(100,DYN_P) on a CX1-PREF problem with 8 orbits.

Objective	ASPEN Wt. Sum	POS found by ASPEN+PART(100,DYN_P)									
		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

The partitioning approach presented is important for reducing the exponential complexity of nonlinear constrained optimization problems. By partitioning a problem into subproblems and by reducing the search space of each partitioned subproblem using our proposed theory, we can reduce the base the exponential complexity of the overall problem. Further, since variable partitioning leads to planning subproblems of similar nature but of small scale, we can exploit existing planners and their efficient pruning techniques to further reduce the search space of these subproblems.

Acknowledgments. This research was supported by the National Aeronautics and Space Administration under Grant NCC 2-1230 and by the National Science Foundation under Grant IIS 03-12084.

References

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

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

[3] Y. X. 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.

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

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

[6] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *J. of Artificial Intelligence Research*, 14:253–302, 2001.

[7] C. Hwang, S. Paidy, and K. Yoon. Mathematical programming with multiple objective: a tutorial. *Computers and Operations Research*, pages 5–31, 1980.

[8] A. K. Jónsson, P. H. Morris, N. Muscettola, and K. Rajan. Planning in interplanetary space: Theory and practice. In *Proc. National Conf. on Artificial Intelligence*. AAAI, 2000.

[9] H. Kautz and B. Selman. Pushing the envelope: planning, propositional logic, and stochastic search. *Proc. Int'l Conf. on AI Planning and Scheduling (AIPS)*, pages 1194–1201, 1996.

[10] H. Kautz and B. Selman. Unifying SAT-based and graph-based planning. In *Proc. Int'l Joint Conf. on Artificial Intelligence*. IJCAI, 1999.

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

[12] F. Lin. A planner called R. *AI Magazine*, pages 73–76, 2001.

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

[14] R. S. Nigenda, X. Nguyen, and S. Kambhampati. AltAlt: Combining the advantages of Graphplan and heuristic state search. Technical report, Arizona State University, 2000.

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

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

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

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

[19] I. Refanidis and I. Vlahavas. The MO-GRT system: Heuristic planning with multiple criteria. *AIPS-02 Workshop on Planning and Scheduling with Multiple Criteria*, 2002.

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

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

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

[23] Z. Wu. *The Theory and Applications of Nonlinear Constrained Optimization using Lagrange Multipliers*. Ph.D. Thesis, Dept. of Computer Science, Univ. of Illinois, Urbana, IL, May 2001.