Partitioning of Temporal Planning Problems in Mixed Space using the Theory of Extended Saddle Points*

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
{wah,chen}@manip.crhc.uiuc.edu

Abstract

We study the partitioning of temporal planning problems formulated as mixed-integer nonlinear programming problems, develop methods to reduce the search space of partitioned subproblems, and propose algorithms for resolving unsatisfied global constraints. The algorithms are based on the necessary and sufficient extended saddle-point condition for constrained local minimization developed in this paper. When compared with the MIPS planner in solving some PDDL2.1 planning problems, our distributed implementation of MIPS shows significant improvements in time and quality.

1. Introduction

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

Many languages and schemes have been developed to represent planning problems, such as STRIPS [6], PDDL [7], and planner-specific languages [4]. In this paper, we formulate a planning problem as a *mixed-integer nonlinear programming* (MINLP) problem.

Our formulation assumes that the continuous time 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 variable vector $z \in Z$ of the problem into $N+1$ subvectors $z(0), \ldots, z(N)$, where $z(t) = (z_1(t), \ldots, z_{u_t}(t))^T$ is a u_t-element *state vector* in mixed space and stage t, and $z_i(t)$ is the i^{th} *dynamic state variable*. The MINLP formulation is as follows:

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

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

Partitioning is possible in temporal planning problems because many of their constraints and objectives are related to activities with temporal locality.

There are two reasons for partitioning the variables of a planning problem into disjoint subsets before solving the problem. First, when the search space of the problem is exponential in size, we can reduce the base of its exponential complexity by reducing the search space of each partitioned subproblem beforehand. As a simple illustration, consider a problem that has been partitioned into $N+1$ stages, each of which has S local states and s of these states satisfy the local constraints. Without resolving the local constraints ahead of time, the search space of the partitioned problem has a worst-case complexity of $O(S^{N+1})$. In contrast, by first resolving the local constraints, the search space is reduced to s^{N+1} possible paths. Although an additional overhead is needed to resolve the local constraints in each stage ($O(S)$ worst-

* Research supported by the National Aeronautics and Space Administration Grant NCC 2-1230 and the National Science Foundation Grant ITR 03-12084.
Proc. IEEE International Conference on Tools with Artificial Intelligence, 2003.

case complexity), it is obvious that there will be a significant reduction in complexity when $s \ll S$. In this paper, we develop new necessary conditions and efficient implementations to further reduce the space in each partition. Second, variable partitioning leads to smaller planning subproblems of similar nature. 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, can be employed to further reduce the search space of these subproblems.

Although there are significant benefits in partitioning, existing planning algorithms in artificial intelligence and optimization techniques in mathematical programming do not exploit partitioning except in some special cases, such as problems with partitionable convex subproblems. The difficulty of partitioning in general is that there are no good methods for resolving global constraints after the partitioned subproblems have been solved. Dynamic programming cannot be applied in temporal planning with global constraints because 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.

In this paper, we develop strategies to partition the search space of temporal planning problems and evaluate algorithms for resolving unsatisfied global constraints. We first summarize in Section 2 the limitations of existing planning techniques in artificial intelligence and search methods in mathematical programming. We present in Section 3 the necessary and sufficient extended saddle-point condition for constrained local minimization in mixed space. Since the condition is true for all Lagrange multipliers larger than some critical multipliers, we propose an iterative approach to look for Lagrange multipliers that are larger than the critical multipliers and to look for variable assignments that minimize the Lagrangian function. In Section 4, by choosing a suitable neighborhood in mixed space, we further reduce the search into stages in order to look for distributed saddle points in each stage, and we resolve global constraints across multiple stages in an outer loop by choosing suitably large Lagrange multipliers. Finally, we demonstrate in Section 5 our approach by using the MIPS planner to solve partitioned subproblems and show significant improvements on some benchmarks.

2. Previous Work

In this section, we summarize the shortcomings of some existing work related to AI planning and Lagrangian methods in optimization.

2.1. Temporal Planning Methods

Existing AI planning and scheduling methods can be classified based on the their state and temporal representations and the search techniques used.

Discrete-time discrete-state methods consist of systematic searches, heuristic searches, local searches, and transformation methods.

Systematic searches that explore the entire state space are complete solvers. Examples include UCPOP, Graphplan, PropPlan, and System R. Systematic solvers are not amenable to variable partitioning because locally feasible or optimal plans in partitioned variable space may not satisfy all global constraints.

Local searches employ heuristic guidance functions to search in discrete path space. Examples include HSP, FF, AltAlt, GRT, and ASPEN. Heuristic solvers do not work well with partitioned plans because their guidance heuristics are generally computed over the entire time horizon in order to estimate the distance from a state to the goal state.

Transformation methods convert a problem into a constrained optimization or satisfaction problem before solving it by existing SAT and ILP solvers. Examples include SATPLAN, Blackbox, and ILP-PLAN. They are not amenable to variable partitioning because they rely on solvers that do not support partitioning.

Discrete-time mixed-state methods employ systematic searches, heuristic searches, and transformation methods. Examples include SIPE-2, O-Plan2, Metric-FF, GRT-R, and LPSAT. The search methods employed by these planners are not amenable to partitioning for reasons similar to those discussed above.

Continuous-time mixed-state methods can be classified into systematic, heuristic, and local searches. Examples include LPG, MIPS, Sapa, ZENO, SHOP2, TALplanner, and Europa. The methods in these planners rely on global information in their search and do not have mechanisms to combine the solutions of partitioned subproblems into global solutions.

2.2. Mathematical Programming Methods

In this section, we survey existing techniques on continuous and mixed-integer optimization and discuss their limitations with respect to partitioning.

Continuous nonlinear programming (CNLP) methods. We examine two sets of conditions used in existing methods for solving CNLPs. Consider the following CNLP with continuous and differentiable functions f, $h = (h_1, \ldots, h_m)^T$ and $g = (g_1, \ldots, g_r)^T$:

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

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

Definition 1. Point x^* is a CLM_c, a constrained local minimum in continuous neighborhood of P_c, if x^* is feasible and $f(x^*) \leq f(x)$ for all feasible $x \in \mathcal{N}_c(x^*)$.

Based on Lagrange-multiplier vectors $\lambda = (\lambda_1, \ldots, \lambda_m)^T \in \mathcal{R}^m$ and $\mu = (\mu_1, \ldots, \mu_r)^T \in \mathcal{R}^r$, the Lagrangian function of P_c is defined as:

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

a) *Karush-Kuhn-Tucker (KKT) necessary condition* [2]. Assuming x^* is a CLM_c and a regular point,[1] then there exist unique λ^* and μ^* such that:

$$\nabla_x L(x^*, \lambda^*, \mu^*) = 0, \tag{4}$$

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

The unique λ and μ that satisfy (4) can be found by solving a system of nonlinear equations in λ, μ, and x. As an illustration, consider P_c with only equality constraints. By specializing KKT, we obtain a system of $n + m$ equations in $n + m$ unknowns:

$$F(x, \lambda) = \begin{bmatrix} \nabla f(x) + B(x)^T \lambda \\ h(x) \end{bmatrix} = 0, \tag{5}$$

where $B(x)^T = [\nabla h_1(x), \ldots, \nabla h_m(x)]$ is the Jacobian of the constraints. In general, a solution method solving (5) needs to involve all the x and λ variables and does not work with multiple subproblems with a partitioned variable space.

b) *Sufficient saddle-point condition* [8, 1]. The concept of saddle points has been studied extensively in the literature. Here, x^* is a saddle point of P_c if there exist λ^* and μ^* such that:

$$L(x^*, \lambda, \mu) \leq L(x^*, \lambda^*, \mu^*) \leq L(x, \lambda^*, \mu^*) \tag{6}$$

for all x satisfying $\|x - x^*\| < \epsilon$ and all $\lambda \in \mathcal{R}^m$ and $\mu \in \mathcal{R}^r$. This condition is only sufficient but not necessary; that is, if x^* is a saddle point, then x^* is a CLM_c, but the converse is not true. Hence, if x^* is a CLM_c, there may not exist λ^* and μ^* that satisfy (6).

The existing saddle-point condition is also not amenable to partitioning because it requires solving for unique λ^* and μ^*. Moreover, it is only a sufficient condition and does not cover all possible CLM_c.

1 Point x is said to be a *regular point* [9] with respect to h if gradient vectors $\nabla h_1(x), \ldots, \nabla h_m(x)$ at x are linearly independent.

c) *Penalty formulations.* A *static-penalty approach* [10] transforms P_c into an unconstrained minimization with the following objective function:

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

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

A *dynamic-penalty approach* [10] increases penalties gradually and solves a sequence of unconstrained problems. The requirement of finding a global optimum of $L_\rho(x, \gamma, \psi)$ for each unconstrained problem may be computationally expensive in practice.

Mixed-integer NLP (MINLP) methods generally decompose a MINLP into subproblems in such a way that, after fixing a subset of the variables, each resulting subproblem is convex and is easily solvable, or can be relaxed and be approximated. There are several types of these algorithms, including generalized Benders decomposition (GBD), outer approximation (OA), generalized cross decomposition (GCD) and branch-and-reduce methods. All those methods require the functions of subproblems to be convex or factorable.

In this paper, we focus on the resolution of global constraints after decomposing a large problem into subproblems. Since our approach does not require the decomposed subproblems to be convex or factorable, it is more general than existing approaches.

3. The Necessary and Sufficient Extended Saddle-Point Condition

In this section, we present our *extended saddle-point condition* (ESPC) in mixed space. By using a new Lagrangian function with transformed constraints, we show that ESPC is necessary as well as sufficient and is satisfied for a range of Lagrange multipliers. The latter property is important because it allows any algorithm implementing ESPC to be partitioned.

3.1. ESPC for Mixed Optimization

Consider the MINLP whose f, g and h are continuous and differentiable functions with respect to the con-

tinuous subspace x:

$$(P_m): \quad \min_{x,y} \quad f(x,y), \;\; x \in \mathcal{R}^v \text{ and } y \in \mathcal{D}^w \quad (7)$$
$$\text{subject to} \;\; h(x,y) = 0 \;\text{ and }\; g(x,y) \leq 0.$$

The goal of solving P_m is to find a constrained local minimum (x, y) with respect to $\mathcal{N}_m(x, y)$, the mixed neighborhood of (x, y). As a discrete neighborhood is a user-defined concept, a mixed neighborhood is a user-defined concept as well. In our theory, we use the following definitions.

Definition 2. A user-defined *discrete neighborhood* $\mathcal{N}_d(y)$ of y in discrete space $\mathcal{D}^w$ is a *finite* user-defined set of points $\{y' \in \mathcal{D}^w\}$ in such a way that y' is reachable from y in one step, that $y' \in \mathcal{N}_d(y) \iff y \in \mathcal{N}_d(y')$, and that it is possible to reach every y'' from any y in one or more steps through neighboring points.

Definition 3. A user-defined *mixed neighborhood* $\mathcal{N}_m(x, y)$ in mixed space $\mathcal{R}^v \times \mathcal{D}^w$ is:

$$\mathcal{N}_m(x,y) = \left\{(x', y) \mid x' \in \mathcal{N}_c(x)\right\} \bigcup \left\{(x, y') \mid y' \in \mathcal{N}_d(y)\right\} \quad (8)$$

Definition 4. Point (x^*, y^*) is a *constrained local minimum in mixed neighborhood* (CLM_m) of P_m if (x^*, y^*) is feasible and $f(x^*, y^*) \leq f(x, y)$ for all feasible $(x, y) \in \mathcal{N}_m(x^*, y^*)$.

Definition 5. The transformed Lagrangian function for P_m in (7) is defined as follows:

$$L_m(x,y,\alpha,\beta) = f(x,y) + \alpha^T|h(x,y)| + \beta^T \max(0, g(x,y)). \quad (9)$$

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

$$L_m(x^*,y^*,\alpha,\beta) \leq L_m(x^*,y^*,\alpha^{**},\beta^{**}) \leq L_m(x,y,\alpha^{**},\beta^{**}) \quad (10)$$

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

The following corollary facilitates the implementation of ESPC in (10) and follows directly from the definition of $\mathcal{N}_m(x, y)$. This definition allows (10) to be partitioned into two independent necessary conditions. Note that such partitioning cannot be accomplished if a mixed neighborhood like $\mathcal{N}_c(x) \times \mathcal{N}_d(y)$ were used.

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

Figure 1. Pseudo code showing the implementation of ESPC for assumed $\bar{\alpha}^*$ and $\bar{\beta}^*$.

Corollary 1. Given $\mathcal{N}_m(x, y)$ defined in (8), the ESPC in (10) can be rewritten as three necessary conditions that, collectively, are necessary and sufficient:

$$L_m(x^*,y^*,\alpha,\beta) \leq L_m(x^*,y^*,\alpha^{**},\beta^{**}) \quad (11)$$
$$L_m(x^*,y^*,\alpha^{**},\beta^{**}) \leq L_m(x^*,y,\alpha^{**},\beta^{**}) \quad (12)$$
$$L_m(x^*,y^*,\alpha^{**},\beta^{**}) \leq L_m(x,y^*,\alpha^{**},\beta^{**}) \quad (13)$$

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

3.2. Implementation Considerations

An important feature of ESPC over the original saddle-point condition in (6) is that, instead of finding unique λ^* and μ^* that minimize $L(x, \lambda^*, \mu^*)$ at x^*, it suffices to minimize $L_m(x, y, \alpha^{**}, \beta^{**})$ by finding any $\alpha^{**} > \alpha^*$ and $\beta^{**} > \beta^*$. Such a property reduces the implementation to an iterative search, instead of solving a system of equations.

Figure 1 shows the pseudo code which solves P_m by looking for x^*, y^*, α^{**}, and β^{**} that satisfy (11)-(13). By performing separate descents of $L_m(x, y, \alpha, \beta)$ in the continuous and discrete neighborhoods in the two inner loops, it looks for a local minimum (x^*, y^*) of $L_m(x, y, \alpha, \beta)$ with respect to points in $\mathcal{N}_m(x, y)$ that satisfy (12) and (13). The outer loop performs ascents for unsatisfied global constraints and stops when a CLM_m has been found.

Despite the simplicity of the implementations, there are three considerations to need to be noted.

First, the algorithms do not prescribe how large α and β should be increased. Obviously, suitable upper bounds on α and β need to be chosen in practice. For the same reason as in dynamic penalty methods described in Section 2.2, α and β should be increased gradually in order to help the search escape from local minima of the Lagrangian function. Once α and β reach the prescribed upper bounds, they may need to be scaled down and the search repeated. The reason for reducing α and β in this

case is to allow the search to move to a different region of the Lagrangian function, in case that it is stuck in an infeasible local minimum of the constrained model.

Second, the number of CLM_m in the search space of a problem depends on the size of the neighborhood chosen. It is clear that if the neighborhood of each point is the entire search space itself, then any CLM_m found is also a constrained global minimum, and the number of CLM_m in the search space is the smallest. In practice, we choose a neighborhood that allows a search to be partitioned into those of continuous and discrete subspaces. A neighborhood like $\mathcal{N}_c(x) \times \mathcal{N}_d(y)$ is not a good choice because it is not partitionable.

Third, the pseudo code in Figure 1 does not specify how descents in the inner loop are to be performed. In solving problems with closed-form continuous and differentiable functions, existing descent procedures, like Newton's search, will need to be modified to account for the discontinuities at $|h(x,y)| = 0$ and $\max(0, g(x,y)) = 0$. In temporal planning problems, since their functions may not be in closed form, heuristic descent procedures can be used to look for local minima of the Lagrangian function. Probes in a search space can be generated based on deterministic, probabilistic, or genetic mechanisms and be accepted based on deterministic or stochastic criteria. For example, the stochastic constrained simulated annealing (CSA) algorithm [11] generates new probes randomly in one of the variables, accepts them based on the Metropolis probability when L_m increases along the x or y dimension and decreases along the α or β dimension, and stops updating α and β when all the constraints are satisfied.

4. Partitioning of the ESPC

In this section, we partition the ESPC presented in the last section into a set of conditions that collectively are necessary and sufficient. By defining a partitionable neighborhood for planning problem P_T, we show that the search for CLM_m can be reduced to finding local saddle points in each stage of P_T and to the resolution of unsatisfied global constraints by choosing appropriate Lagrange multipliers. Finally, we show an iterative implementation of the partitioned conditions.

4.1. Necessary and Sufficient ESPC for Partitioned Problems

The goal of solving P_T in (1) is to find a plan z that is a CLM_m with respect to its mixed neighborhood $\mathcal{N}_m(z)$. To simplify our discussion, we do not partition z into discrete and continuous parts in the following derivation, although it is understood that each partition will need to be further decomposed in the same way as in Figure 1. To enable the partitioning of ESPC into independent necessary conditions, we define the neighborhood of plan z as follows:

Definition 6. $\mathcal{N}_p(z)$, the *partitionable mixed neighborhood of plan* z, is defined as:

$$\mathcal{N}_p(z) = \bigcup_{t=0}^{N} \mathcal{N}_p^{(t)}(z) = \bigcup_{t=0}^{N} \Big\{ z' \;\Big|\; z'(t) \in \mathcal{N}_m(z(t)) \text{ and } z'(i \mid i \neq t) = z(i) \Big\}, \quad (14)$$

where $\mathcal{N}_m(z(t))$ is the mixed-space neighborhood of state vector $z(t)$ in stage t.

Intuitively, $\mathcal{N}_p(z)$ is partitioned into $N+1$ disjoint sets of neighborhoods, each perturbing z in one of the stages of P_T. By considering P_T in (1) as an MINLP, we can apply (9) and Theorem 1 to get the ESPC conditions. Based on the partitionable neighborhood, these conditions can be further partitioned into a set of distributed conditions.

Definition 7. The transformed Lagrangian function for P_T in (1) is defined as follows:

$$L_m(z,\alpha,\beta,\gamma,\eta) = J(z) + \sum_{t=0}^{N} \Big\{ \alpha(t)^T |h^{(t)}(z(t))| + \beta(t)^T \max(0, g^{(t)}(z(t))) \Big\} + \gamma^T |H(z)| + \eta^T \max(0, G(z)), \quad (15)$$

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 Lagrange multipliers.

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

$$\Gamma_m(z^*, \alpha(t), \beta(t), \gamma^{**}, \eta^{**}) \leq \Gamma_m(z^*, \alpha(t)^{**}, \beta(t)^{**}, \gamma^{**}, \eta^{**}), \quad (16)$$

$$\Gamma_m(z^*, \alpha(t)^{**}, \beta(t)^{**}, \gamma^{**}, \eta^{**}) \leq \Gamma_m(z, \alpha(t)^{**}, \beta(t)^{**}, \gamma^{**}, \eta^{**}), \quad (17)$$

$$L_m(z^*, \alpha^{**}, \beta^{**}, \gamma, \eta) \leq L_m(z^*, \alpha^{**}, \beta^{**}, \gamma^{**}, \eta^{**}), \quad (18)$$

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

$$\Gamma_m(z,\alpha(t),\beta(t),\gamma,\eta) = J(z) + \alpha(t)^T |h^{(t)}(z(t))| + \beta(t)^T \max(0, g^{(t)}(z(t))) + \gamma^T |H(z)| + \eta^T \max(0, G(z)). \quad (19)$$

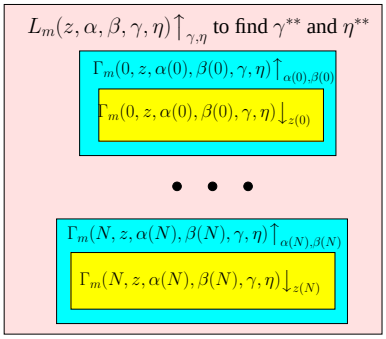


Figure 2. An iterative implementation of the partitioned necessary and sufficient ESPC in (16)-(18) to look for CLM_m of P_T.

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

4.2. Implementation Considerations

Figure 2 illustrates an iterative implementation of the conditions in Theorem 2.

In the two inner nested loops of stage t, the algorithm looks for a local saddle point of $\Gamma_m(t,z,\alpha(t),\beta(t),\gamma,\eta)$. This is done by updating z and Lagrange multipliers $\alpha(t)$ and $\beta(t)$ associated with the local constraints, using fixed γ and η in the global constraints. With fixed γ and η, the algorithm is actually finding $z(t)$ that solves the following MINLP in stage t:

$$\begin{aligned} \min_{z(t)} \quad & J(z)+\gamma^T H(z)+\eta^T G(z) \qquad (20)\\ \text{subject to} \quad & h(t,z(t))=0 \quad \text{and} \quad g(t,z(t))\le 0. \end{aligned}$$

Since this is a well-defined MINLP, any existing solver can be used to solve it with little modification. As illustrated in the next section, we use the MIPS planner to solve planning subproblems, each defined by the objective and the local constraints in (20).

After all the local searches have been performed, the penalties on unsatisfied global constraints are increased in an outer loop. The search terminates when a feasible local minimum in the constrained model is found.

It is important to point out that the traditional Lagrangian theory cannot support the iterative search described above. In the traditional theory, each stage related to a global constraint will require a unique Lagrange-multiplier value for this constraint. Since possibly different multiplier values may be associated with a global constraint in different stages, an iterative search will have difficulty to converge to a single multiplier value for each global constraint.

5. A Distributed Implementation of MIPS

In this section, we describe briefly the algorithms used in the mixed-space MIPS planner, the PDDL2.1 benchmarks tested, and our experimental results. For comparison, results on the application of our approach on the discrete-space ASPEN planner [4] has been reported elsewhere [3].

MIPS [5] is a heuristic anytime planner that performs static analysis of a problem instance in mixed space and continuous time, searches for an optimized sequential plan, and performs a critical path analysis called PERT to generate optimal parallel plans from a sequence of operators and their precedence relations. Using a weighted A^* algorithm, it finds an optimal feasible path from initial state s_i to goal state $s_g \in \mathcal{G}$ in a state space of propositional facts and numeric variables. It can also optimize an arbitrary objective by incorporating the objective in the heuristic evaluation.

MIPS can handle the STRIPS subset of the PDDL language and can cope with numeric quantities and durations in PDDL 2.1 [7]. We use MIPS in our experiments because it performs well on the PDDL2.1 benchmarks and its source code is readily available.

5.1. Implementation of Distributed Search

Figure 3 shows the MIPS+DIS algorithm that generates an initial (possibly infeasible) plan of a planning problem, formulates the problem in a Lagrangian function, decomposes the states into multiple stages, solves each subproblem locally, and resolves unsatisfied global constraints by increasing their Lagrange multipliers.

In a problem solved by MIPS, state s with n_f facts and n_r numerical variables is specified as $s=(s_f,s_r)$, where s_f lists the true facts at s, and s_r is an n_r-vector of instantiated values of the variables. The set of grounded facts are further partitioned into *symmetry groups* in the static-analysis phase in such a way that each element of s_f is a fact from a unique symmetry group.

After partitioning, the local planning problem in stage t has initial state $s_i(t)$ and goal state $s_i(t+1)$ (see

1. **procedure MIPS+DIS**
2. generate initial plan using relaxed operators;
3. **repeat**
4. $iter \leftarrow 0$;
5. **for** $t = 0$ **to** N
6. $num\_trials \leftarrow 0$;
7. **repeat**
8. $num\_trials \leftarrow num\_trials + 1$;
9. generate an initial state in $\mathcal{N}_m(s_i(t))$ for stage t;
10. call MIPS to solve the subproblem in stage t;
11. evaluate $\Gamma_m(t)$ of the solution plan from MIPS;
12. **until** ($\Gamma_m(t)$ is improved) **or** ($num\_trials > max\_trials$);
13. **end_for**
14. update Lagrange multipliers γ on unsatisfied global constraints;
15. $iter \leftarrow iter + 1$;
16. **if** ($iter$ % τ == 0) dynamically repartition the stages;
17. **until** no change in z and γ in an iteration;
18. **end_procedure**

Figure 3. MIPS+DIS: A distributed iterative procedure using MIPS to find points that satisfy Theorem 2.

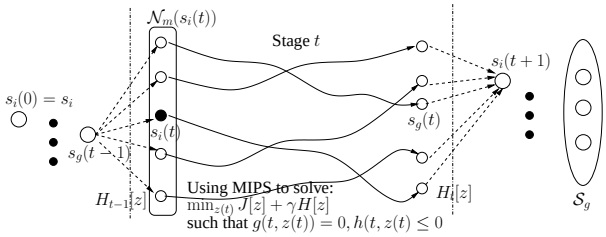


Figure 4. An illustration of the distributed search in MIPS+DIS.

Figure 4 for the states defined in stage t). Since the initial local plan may not be feasible, we need to define a new local planning problem that, when solved, will hopefully make the overall planning problem feasible.

We define the *neighborhood* $\mathcal{N}_m(s)$ of s to include s as well as the set of states that differ from s by at most one fact and that the different facts of two neighboring states exist in a symmetry group. To generate a neighboring state from s, we randomly pick a fact in s and perturb the fact to a different fact in the corresponding symmetry group. Note that, since s_r, the numeric part of s, is not changed in the process, there may not exist a valid action for a transition from s to its neighbor.

We measure the *distance* $D(s,t)$ between two states $s = (s_f, s_r)$ and $t = (t_f, t_r)$ as the sum of N_d (the number of different facts in s and t) and the normalized difference between their numerical parts:

$$D(s,t) = N_d + \sum_{i=1}^{n_r} \frac{|s_{r_i} - t_{r_i}|}{\max(s_{r_i}, t_{r_i})}. \quad (21)$$

Hence, $D(s,t) = 0$ iff s and t are identical states. Next, we define $\mathcal{S}(s)$, the set of *successor* (different from neighborhood) states of s, in such a way that there exists a valid action that brings s to v for all $v \in \mathcal{S}(s)$. Last, we define the *transition distance* $T(s,t)$ as the minimum distance between s and t over all successors of s:

$$T(s,t) = \min_{v \in \mathcal{S}(s)} D(v,t). \quad (22)$$

According to this definition, $T(s,t) = 0$ if there exists a valid action to bring s to t.

We are now ready to define the new local planning problem in stage t for MIPS to solve (Line 10 of Figure 3). The problem has the same domain specification as in the overall problem, initial state $s'_i(t) \in \mathcal{N}_m(s_i(t))$, and goal state $s_i(t+1)$. In addition, there are two global constraints at the boundaries between stage t and the predecessor and successor stages:

$$H_{t-1}(z) = T(s_g(t-1), s_i(t)) = 0; \quad (23)$$
$$H_t(z) = T(s_g(t), s_i(t+1)) = 0. \quad (24)$$

Hence, $H_{t-1}(z) = 0$ (*resp.* $H_t(z) = 0$) is satisfied if and only if there is a valid action to bring $s_g(t-1)$ (*resp.* $s_g(t)$) to $s_i(t)$ (*resp.* $s_i(t+1)$). We also define the objective of the local problem:

$$f(t) = J(z) + \gamma_{t-1} H_{t-1}(z) + \gamma_t H_t(z), \quad (25)$$

where γ_{t-1} and γ_t are fixed Lagrange multipliers associated with the two global constraints.

After solving the local problem defined, MIPS returns an optimal feasible plan from $s'_i(t)$ to $s_i(t+1)$ if one exists; otherwise, it returns a feasible plan from $s'_i(t)$ to $s_g(t)$ that minimizes $f(t)$. We accept this plan if it improves $f(t)$; otherwise, we repeat generating new initial states in $\mathcal{N}_m(s_i(t))$ until we find a better plan or when the maximum number of trials is exceeded (Line 12). In our experiments, we set max_trials to 5.

After completing the $N+1$ local subproblems in an iteration (Line 14), we update the Lagrange multipliers of all unsatisfied global constraints, using $\omega > 0$ to control the rate of increase of γ_t:

$$\gamma_t \leftarrow \gamma_t + \omega \times H_t(z), \quad t = 0, 1, \ldots, N. \quad (26)$$

We set $\omega = 0.01 J_a$, where J_a is the average value of $J(z)$ in the last three iterations.

We repartition the stages dynamically by adjusting the boundary of stages every certain number (τ in Figure 3) of iterations. This is accomplished by counting

the number of state transitions from s_i to s_g at the end of the outer loop (Line 16) and redefine the stage boundaries in order for each stage to have approximately the same number of state transitions. As a result of the repartitioning, the number of violated global constraints in a stage may be different from one. In our experiments, we set $N = 20$ and $\tau = 5$.

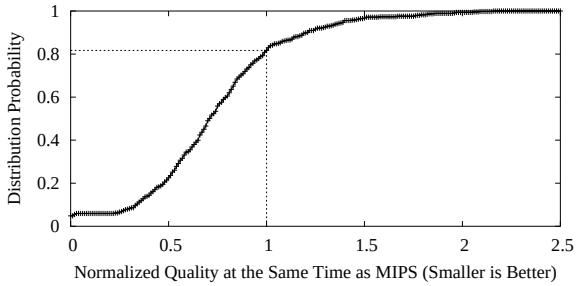
5.2. Experimental Results

We show that our MIPS+DIS planner improves significantly over the original MIPS planner on a set of PDDL2.1 planning benchmarks used in the Third International Planning Competition. The problems studied belong to a number of domains, including $DepotNumeric$, $DepotSim$, $DepotTime$, $DriveLogNumeric$, $DriveLogSim$, $DriveLogTime$, $ZenoTravelNumeric$, $ZenoTravelSim$, and $ZenoTravelTime$.

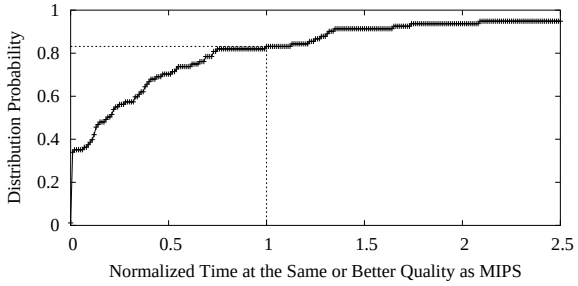
In our experiments on MIPS, we used the most recent executables downloaded from its Web site and ran it with default parameters and a maximum time limit of 10^6ms. All experiments were conducted on an AMD Athlon MP2000 PC with Linux Redhat 7.2.

For the 120 (out of a total of 160) problems solvable by MIPS, Figure 5a plots the distribution of the quality of the solution found by MIPS+DIS normalized with respect to that of MIPS, using the same amount of time taken by MIPS to find the solution. Similarly, Figure 5b plots the distribution of the time taken by MIPS+DIS to find a solution of the same or better quality as obtained by MIPS, normalized with respect to the time taken by MIPS. The graphs do not include the results on 30 problems for which MIPS+DIS can solve but MIPS cannot find any feasible plan in 10^6ms. The results show that MIPS+DIS is able to improve over MIPS in 81.7% of the cases in quality or 83.2% in time.

Of the 150 of the 160 problems solvable by MIPS+DIS, MIPS+DIS can find a feasible solution faster than MIPS in 94.4% of the cases and a better final solution quality in 93.8% of the cases.



a) Distribution of the quality of the solution found by MIPS+DIS normalized with respect to that of MIPS, using the same amount of time taken by MIPS to find the solution



b) Distribution of the time taken by MIPS+DIS to find a solution of the same or better quality as obtained by MIPS, normalized with respect to the time taken by MIPS

Figure 5. Normalized time and quality of MIPS+DIS with respect to MIPS on the 120 problems solvable by MIPS (out of a total of 160 problems). The time and quality of MIPS is (1,1) for all these problems.

References

[1] M. Avriel. *Nonlinear Programming: Analysis and Methods*. Prentice-Hall, Inc., Englewood Cliffs, NJ., 1976.

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

[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] S. Edelkamp. Mixed propositional and numerical planning in the model checking integrated planning system. *Proc. Int'l Conf. on AI Planning and Scheduling (AIPS)*, 2002.

[6] R. E. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3 & 4):189–208, 1971.

[7] M. Fox and D. Long. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Tech. Rep., Dept. of Computer Science, Univ. of Durham, Durhan, UK*, February 2002.

[8] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In J. Neyman, editor, *Proc. Second Berkeley Sympos. Math. Stat. Prob.*, pages 481–492. University of California Press, 1951.

[9] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, Reading, MA, 1984.

[10] R. L. Rardin. *Optimization in Operations Research*. Prentice Hall, 1998.

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