Theory and Applications of Simulated Annealing for Nonlinear Constrained Optimization[1]

Benjamin W. Wah[1], Yixin Chen[2] and Tao Wang[3]
[1]*Department of Electrical and Computer Engineering and the Coordinated Science Laboratory, University of Illinois,*
[2]*Department of Computer Science, Washington University,*
[3]*Synopsys, Inc.*
USA

1. Introduction

A general *mixed-integer nonlinear programming problem* (MINLP) is formulated as follows:

$$(P_m): \qquad \min_z \quad f(z), \qquad \text{subject to} \quad h(z) = 0 \quad \text{and} \quad g(z) \le 0, \tag{1}$$

where $z = (x, y)^T \in Z$; $x \in \mathbb{R}^v$ and $y \in \mathbb{D}^w$ are, respectively, bounded continuous and discrete variables; $f(z)$ is a lower-bounded objective function; $g(z) = (g_1(z), \ldots, g_r(z))^T$ is a vector of r inequality constraint functions;[2] and $h(z) = (h_1(z), \ldots, h_m(z))^T$ is a vector of m equality constraint functions. Functions $f(z)$, $g(z)$, and $h(z)$ are general functions that can be discontinuous, non-differentiable, and not in closed form.
Without loss of generality, we present our results with respect to minimization problems, knowing that maximization problems can be converted to minimization ones by negating their objectives. Because there is no closed-form solution to P_m, we develop in this chapter efficient procedures for finding locally optimal and feasible solutions to P_m, demonstrate that our procedures can lead to better solutions than existing methods, and illustrate the procedures on two applications. The proofs that our procedures have well-behaved convergence properties can be found in the reference [27], We first define the following terms.

[1] Research supported by the National Science Foundation Grant IIS 03-12084 and a Department of Energy Early Career Principal Investigator Grant.
[2] Given two vectors V_1 and V_2 of the same dimension, $V_1 \ge V_2$ means that each element of V_1 is greater than or equal to the corresponding element of V_2; $V_1 > V_2$ means that at least one element of V_1 is greater than the corresponding element of V_2 and the other elements are greater than or equal to the corresponding elements of V_2.

Source: Simulated Annealing, Book edited by: Cher Ming Tan, ISBN 978-953-7619-07-7, pp. 420, February 2008, I-Tech Education and Publishing, Vienna, Austria

Definition 1. A *mixed neighborhood* $\mathcal{N}_m(z)$ for $z = (x, y)^T$ in the mixed space $\mathbb{R}^v \times \mathbb{D}^w$ is:

$$\mathcal{N}_m(z) = \Big\{(x', y)^T \mid x' \in \mathcal{N}_c(x)\Big\} \cup \Big\{(x, y')^T \mid y' \in \mathcal{N}_d(y)\Big\}, \tag{2}$$

where $\mathcal{N}_c(x) = \{x' : \|x' - x\| \le \epsilon$ and $\epsilon \to 0\}$ is the *continuous neighborhood* of x, and the *discrete neighborhood* $\mathcal{N}_d(y)$ is a *finite* user-defined set of points $\{y' \in \mathbb{D}^w\}$ such that $y' \in \mathcal{N}_d(y) \Leftrightarrow y \in \mathcal{N}_d(y')$ [1]. Here, $\epsilon \to 0$ means that ϵ is arbitrarily close to 0.

Definition 2. Point z of P_m is a *feasible point* iff $h(z) = 0$ and $g(z) \le 0$.

Definition 3. Point z^* is a *constrained local minimum* (CLM_m) of P_m iff z^* is feasible, and $f(z^*) \le f(z)$ with respect to all feasible $z \in \mathcal{N}_m(z^*)$.

Definition 4. Point z^* is a *constrained global minimum* (CGM_m) of P_m iff z^* is feasible, and $f(z^*) \le f(z)$ for every feasible $z \in \mathcal{Z}$. The set of all CGM_m of P_m is $\mathcal{Z}_{\text{opt}}$.

Note that a discrete neighborhood is a user-defined concept because it does not have any generally accepted definition. Hence, it is possible for $z = (x, y)^T$ to be a CLM_m to a neighborhood $\mathcal{N}_d(y)$ but not to another neighborhood $\mathcal{N}_{d_1}(y)$. The choice, however, does not affect the validity of a search as long as one definition is consistently used throughout. Normally, one may choose $\mathcal{N}_d(y)$ to include discrete points closest to z, although a search will also be correct if the neighborhood includes "distant" points.

Finding a CLM_m of P_m is often challenging. First, $f(z)$, $g(z)$, and $h(z)$ may be non-convex and highly nonlinear, making it difficult to even find a feasible point or a feasible region. Moreover, it is not always useful to keep a search within a feasible region because there may be multiple disconnected feasible regions. To find high-quality solutions, a search may have to move from one feasible region to another. Second, $f(z)$, $g(z)$, and $h(z)$ may be discontinuous or may not be differentiable, rendering it impossible to apply existing theories based on gradients.

A popular method for solving P_m is the penalty method (Section 2.1). It transforms P_m into an unconstrained penalty function and finds suitable penalties in such a way that a global minimum of the penalty function corresponds to a CGM_m of P_m. Because it is computationally intractable to look for global minima when the penalty function is highly nonlinear, penalty methods are only effective for finding CGM_m in special cases.

This chapter is based on the theory of extended saddle points in mixed space [25, 29] (Section 2.2), which shows the one-to-one correspondence between a CLM_m of P_m and an *extended saddle point* (ESP) of the corresponding penalty function. The necessary and sufficient condition allows us to find a CLM_m of P_m by looking for an ESP of the corresponding penalty function.

One way to look for those ESPs is to minimize the penalty function, while gradually increasing its penalties until they are larger than some thresholds. The approach is not sufficient because it also generates stationary points of the penalty function that are not

CLM_m of P_m. To avoid those undesirable stationary points, it is possible to restart the search when such stationary points are reached, or to periodically decrease the penalties in order for the search to escape from such local traps. However, this simple greedy approach for updating penalties may not always work well across different problems.

Our goals in this chapter are to design efficient methods for finding ESPs of a penalty formulation of P_m and to illustrate them on two applications. We have made three contributions in this chapter.

First, we propose in Section 3.1 a constrained simulated annealing algorithm (CSA), an extension of conventional simulated annealing (SA) [18], for solving P_m. In addition to probabilistic descents in the problem-variable subspace as in SA, CSA does probabilistic ascents in the penalty subspace, using a method that controls descents and ascents in a unified fashion. Because CSA is sample-based, it is inefficient for solving large problems. To this end, we propose in Section 3.2 a constraint-partitioned simulated annealing algorithm (CPSA). By exploiting the locality of constraints in many constraint optimization problems, CPSA partitions P_m into multiple loosely coupled subproblems that are related by very few global constraints, solves each subproblem independently, and iteratively resolves the inconsistent global constraints.

Second, we show in Section 4 the asymptotic convergence of CSA and CPSA to a constrained global minimum with probability one in discrete constrained optimization problems, under a specific temperature schedule [27]. The property can be proved by modeling the search as a strongly ergodic Markov chain and by showing that CSA and CPSA minimize an implicit virtual energy at any constrained global minimum with probability one. The result is significant because it extends conventional SA, which guarantees asymptotic convergence in discrete unconstrained optimization, to that in discrete constrained optimization. It also establishes the condition under which optimal solutions can be found in constraint-partitioned nonlinear optimization problems.

Last, we evaluate CSA and CPSA in Section 5 by solving some benchmarks in continuous space and by demonstrating their effectiveness when compared to other dynamic penalty methods. We also apply CSA to solve two real-world applications, one on sensor-network placements and another on out-of-core compiler code generation.

2. Previous work on penalty methods

Direct and penalty methods are two general approaches for solving P_m. Since direct methods are only effective for solving some special cases of P_m, we focus on penalty methods in this chapter.

A penalty function of P_m is a summation of its objective and constraint functions weighted by penalties. Using penalty vectors $\alpha \in \mathbb{R}^m$ and $\beta \in \mathbb{R}^r$, the general penalty function for P_m is:

$$L_p\big((z,\alpha,\beta)^T\big) = f(z) + \alpha^T P(h(z)) + \beta^T Q(g(z)), \tag{3}$$

where P and Q are transformation functions. The goal of a penalty method is to find suitable α^* and β^* in such a way that z^* that minimizes (3) corresponds to either a CLM_m or a

CGM_m of P_m. Penalty methods belong to a general approach that can solve continuous, discrete, and mixed constrained optimization problems, with no continuity, differentiability, and convexity requirements.

When $P(g(z))$ and $Q(h(z))$ are general functions that can take positive and negative values, unique values of α^* and β^* must be found in order for a local minimum z^* of (3) to correspond to a CLM_m or CGM_m of P_m. (The proof is not shown.) However, the approach of solving P_m by finding local minima of (3) does not always work for discrete or mixed problems because there may not exist any feasible penalties at z^*. (This behavior is illustrated in Example 1 in Section 2.1.) It is also possible for the penalties to exist at z^* but (3) is not at a local minimum there. A special case exists in continuous problems when constraint functions are continuous, differentiable, and regular. For those problems, the Karush-Kuhn-Tucker (KKT) condition shows that unique penalties always exist at constrained local minima [21]. In general, existing penalty methods for solving P_m transform $g(z)$ and $h(z)$ in (3) into non-negative functions before finding its local or global minima. In this section, we review some existing penalty methods in the literature.

2.1 Penalty methods for constrained global optimization

Static penalty methods. A *static-penalty method* [21, 22] formulates P_m as the minimization of (3) when its transformed constraints have the following properties: a) $P(h(z)) \geq 0$ and $Q(g(z)) \geq 0$; and b) $P(h(z)) = 0$ iff $h(z) = 0$, and $Q(g(z)) = 0$ iff $g(z) \leq 0$. By finding suitable penalty vectors α and β, an example method looks for z^* by solving the following problem with constant $\rho > 0$:

$$(P_1): \qquad \min_z L_s\big((z,\alpha,\beta)^T\big) = \min_z \left[f(z) + \sum_{i=1}^{m} \alpha_i \, |h_i(z)|^\rho + \sum_{j=1}^{r} \beta_j \, \big(g_j(z)^+\big)^\rho \right], \quad (4)$$

where $g_j(z)^+ = \max(0, g_j(z))$, and $g(z)^+ = (g_1(z)^+, \ldots, g_r(z)^+)^T$.

Given z^*, an interesting property of P_1 is that z^* is a CGM_m of P_m iff there exist finite $\alpha^* \geq 0$ and $\beta^* \geq 0$ such that z^* is a global minimum of $L_s((z, \alpha^{**}, \beta^{**})_T)$ for any $\alpha^{**} > \alpha^*$ and $\beta^{**} > \beta^*$. To show this result, note that α_i and β_j in P_1 must be greater than zero in order to penalize those transformed violated constraint functions $|h_i(z)|^\rho$ and $(g_j(z)^+)^\rho$, which are non-negative with a minimum of zero. As (4) is to be minimized with respect to z, increasing the penalty of a violated constraint to a large enough value will force the corresponding transformed constraint function to achieve the minimum of zero, and such penalties always exist if a feasible solution to P_m exists. At those points where all the constraints are satisfied, every term on the right of (4) except the first is zero, and a global minimum of (4) corresponds to a CGM_m of P_m.

Example 1. Consider the following simple discrete optimization problem:

$$\min_{\substack{y\in\{-3,-2,\\-1,0,1,2\}}} f(y) = \begin{cases} 0 & \text{if } y \geq 0 \\ y & \text{if } y = -1, -2 \\ -4 & \text{if } y = -3 \end{cases} \qquad \text{subject to } y = 0. \tag{5}$$

Obviously, $y^* = 0$. Assuming a penalty function $L_p((y, \alpha)^T) = f(y) + \alpha y$ and $\mathcal{N}_d(y) = \{y-1, y+1\}$, there is no single α^* that can make $L_p((y, \alpha^*)^T)$ a local minimum at $y^* = 0$ with respect to $y = \pm 1$. This is true because we arrive at an inconsistent α^* when we solve the following inequalities:

$$0 = L_p\big((0,\alpha^*)^T\big) \leq \begin{cases} L_p\big((-1,\alpha^*)^T\big) = f(-1) - \alpha^* = -1 - \alpha^* \\ L_p\big((1,\alpha^*)^T\big) = f(1) + \alpha^* = 0 + \alpha^* \end{cases} \Longrightarrow \begin{cases} \alpha^* \leq -1 & \text{when } y = -1 \\ \alpha^* \geq 0 & \text{when } y = 1. \end{cases}$$

On the other hand, by using $L_s((y, \alpha)^T) = f(y) + \alpha\, |y|$ and by setting $\alpha^* = \frac{4}{3}$, the CGM_d of (5) corresponds to the global minimum of $L_s((y, \alpha^{**})^T)$ for any $\alpha^{**} > \alpha^*$. ■

A variation of the static-penalty method proposed in [16] uses discrete penalty values and assigns a penalty value $\alpha_i(h_i(z))$ when $h_i(z)$ exceeds a discrete level ℓ_i (*resp.*, $\beta_j(g_j(z))$ when $g_j(z)^+$ exceeds a discrete level ℓ_j), where a higher level of constraint violation entails a larger penalty value. The penalty method then solves the following minimization problem:

$$(P_2): \quad \min_z L_s\big((z,\alpha,\beta)^T\big) = \min_z \left[f(z) + \sum_{i=1}^{m} \alpha_i(h_i(z))\, h_i^2(z) + \sum_{j=1}^{r} \beta_j(g_j(z)) \big(g_j(z)^+\big)^2 \right]. \tag{6}$$

A limitation common to all static-penalty methods is that their penalties have to be found by trial and error. Each trial is computationally expensive because it involves finding a global minimum of a nonlinear function. To this end, many penalty methods resort to finding local minima of penalty functions. However, such an approach is heuristic because there is no formal property that relates a CLM_m of P_m to a local minimum of the corresponding penalty function. As illustrated earlier, it is possible that no feasible penalties exist in order to have a local minimum at a CLM_m in the penalty function. It is also possible for the penalties to exist at the CLM_m but the penalty function is not at a local minimum there.

Dynamic penalty methods. Instead of finding α^{**} and β^{**} by trial and error, a *dynamic-penalty method* [21, 22] increases the penalties in (4) gradually, finds the global minimum z^* of (4) with respect to z, and stops when z^* is a feasible solution to P_m. To show that z^* is a CGM_m when the algorithm stops, we know that the penalties need to be increased when z^* is a global minimum of (4) but not a feasible solution to P_m. The first time z^* is a feasible solution to P_m, the solution must also be a CGM_m. Hence, the method leads to the smallest

α^{**} and β^{**} that allow a CGM_m to be found. However, it has the same limitation as static-penalty methods because it requires computationally expensive algorithms for finding the global minima of nonlinear functions.

There are many variations of dynamic penalty methods. A well-known one is the *non-stationary method* (NS) [17] that solves a sequence of minimization problems with the following in iteration *t*:

$$(P_3): \qquad \min_z L_t\big((z,\alpha,\beta)^T\big) \;=\; \min_z \left[f(z) + \sum_{i=1}^{m} \alpha_i(t)\; |h_i(z)|^\rho + \sum_{j=1}^{r} \beta_j(t)\big(g_j(z)^+\big)^\rho \right] \tag{7}$$

where $\alpha_i(t+1) = \alpha_i(t) + C \cdot |h_i(z(t))|, \quad \beta_j(t+1) = \beta_j(t) + C \cdot g_j(z(t))^+$.

Here, C and ρ are constant parameters, with a reasonable setting of $C = 0.01$ and $\rho = 2$. An advantage of the NS penalty method is that it requires only a few parameters to be tuned.

Another dynamic penalty method is the *adaptive penalty method* (AP) [5] that makes use of a feedback from the search process. AP solves the following minimization problem in iteration *t*:

$$(P_4): \qquad \min_z L_t\big((z,\alpha,\beta)^T\big) = \min_z \left[f(z) + \sum_{i=1}^{m} \alpha_i(t)\; h_i(z)^2 + \sum_{j=1}^{r} \beta_j(t)\big(g_j(z)^+\big)^2 \right], \tag{8}$$

where $\alpha_i(t)$ is, respectively, increased, decreased, or left unchanged when the constraint $h_i(z) = 0$ is respectively, infeasible, feasible, or neither in the last ℓ iterations. That is,

$$\alpha_i(t+1) = \begin{cases} \frac{\alpha_i(t)}{\lambda_1} & \text{if } h_i(z(i)) = 0 \text{ is feasible in iterations } t-\ell+1,\ldots,t \\ \lambda_2 \cdot \alpha_i(t) & \text{if } h_i(z(i)) = 0 \text{ is infeasible in iterations } t-\ell+1,\ldots,t \\ \alpha_i(t) & \text{otherwise,} \end{cases} \tag{9}$$

where ℓ is a positive integer, $\lambda_1, \lambda_2 > 1$, and $\lambda_1 \neq \lambda_2$ in order to avoid cycles in updates. We use $\ell = 3$, $\lambda_1 = 1.5$, and $\lambda_2 = 1.25$ in our experiments. A similar rule applies to the updates of $\beta_j(t)$.

The *threshold penalty method* estimates and dynamically adjusts a near-feasible threshold $q_i(t)$ (*resp.*, $q'_j(t)$) for each constraint in iteration t. Each threshold indicates a reasonable amount of violation allowed for promising but infeasible points during the solution of the following problem:

$$(P_5): \qquad \min_z L_t\big((z,\alpha,\beta)^T\big) = \min_z \left\{ f(z) + \alpha(t) \left[\sum_{i=1}^{m} \left(\frac{h_i(z)}{q_i(t)} \right)^2 + \sum_{j=1}^{r} \left(\frac{g_j(z)^+}{q'_j(t)} \right)^2 \right] \right\}. \tag{10}$$

There are two other variations of dynamic penalty methods that are not as popular: the death penalty method simply rejects all infeasible individuals [4]; and a penalty method that uses the number of violated constraints instead of the degree of violations in the penalty function [20].

Exact penalty methods. Besides the dynamic penalty methods reviewed above that require solving a series of unconstrained minimization problems under different penalty values, the *exact penalty methods* are another class of penalty methods that can yield an optimal solution by solving a single unconstrained optimization of the penalty function with appropriate penalty values. The most common form solves the following minimization problem in continuous space [35, 6]:

$$\min_x L_e\big((x,c)^T\big) = \min_x \left[f(x) + c\left(\sum_{i=1}^m |h_i(x)| + \sum_{j=1}^r g_j(x)^+\right)\right]. \tag{11}$$

It has been shown that, for continuous and differentiable problems and when certain constraint qualification conditions are satisfied, there exists $c* > 0$ such that the $x*$ that minimizes (11) is also a global optimal solution to the original problem [35, 6]. In fact, c needs to be larger than the summation of all the Lagrange multipliers at $x*$, while the existence of the Lagrange multipliers requires the continuity and differentiability of the functions.

Besides (11), there are various other formulations of exact penalty methods [11, 12, 10, 3]. However, they are limited to continuous and differentiable functions and to global optimization. The theoretical results for these methods were developed by relating their penalties to their Lagrange multipliers, whose existence requires the continuity and differentiability of the constraint functions.

In our experiments, we only evaluate our proposed methods with respect to dynamic penalty methods P_3 and P_4 for the following reasons. It is impractical to implement P_1 because it requires choosing some suitable penalty values a priori. The control of progress in solving P_2 is difficult because it requires tuning many $(\ell \cdot (m+r))$ parameters that are hard to generalize. The method based on solving P_5 is also hard to generalize because it depends on choosing an appropriate sequence of violation thresholds. Reducing the thresholds quickly leads to large penalties and the search trapped at infeasible points, whereas reducing the thresholds slowly leads to slow convergence. We do not evaluate exact penalty methods because they were developed for problems with continuous and differentiable functions.

2.2 Necessary and sufficient conditions on constrained local minimization

We first describe in this section the theory of extended saddle points (ESPs) that shows the one-to-one correspondence between a CLM_m of P_m and an ESP of the penalty function. We then present the partitioning of the ESP condition into multiple necessary conditions and the formulation of the corresponding subproblems. Because the results have been published earlier [25, 29], we only summarize some high-level concepts without the precise formalism and their proofs.

Definition 5. For penalty vectors $\alpha \in \mathbb{R}^m$ and $\beta \in \mathbb{R}^r$, we define a *penalty function* of P_m as:

$$L_m\big((z,\alpha,\beta)^T\big) = f(z) + \alpha^T|h(z)| + \beta^T g(z)^+ = f(z) + \sum_{i=1}^m \alpha_i|h_i(z)| + \sum_{j=1}^r \beta_j g_j(z)^+. \tag{12}$$

Next, we informally define a constraint-qualification condition needed in the main theorem [25]. Consider a feasible point $z' = (x', y')^T$ and a neighboring point $z'' = (x'+\vec{p}, y')^T$ under an infinitely small perturbation along direction $\vec{p} \in X$ in the x subspace. When the *constraint-qualification condition* is satisfied at z', it means that there is no $\vec{p}$ such that the rates of change of all equality and active inequality constraints between z'' and z' are zero. To see why this is necessary, assume that $f(z)$ at z' decreases along $\vec{p}$ and that all equality and active inequality constraints at z' have zero rates of change between z'' and z'. In this case, it is not possible to find some finite penalty values for the constraints at z'' in such a way that leads to a local minimum of the penalty function at z' with respect to z''. Hence, if the above scenario were true for some $\vec{p}$ at z', then it is not possible to have a local minimum of the penalty function at z'. In short, constraint qualification at z' requires at least one equality or active inequality constraint to have a non-zero rate of change along each direction $\vec{p}$ at z' in the x subspace.

Theorem 1. Necessary and sufficient condition on CLM_m of P_m [25]. Assuming $z^* \in Z$ of P_m satisfies the constraint-qualification condition, then z^* is a CLM_m of P_m iff there exist some finite $\alpha^* \geq 0$ and $\beta^* \geq 0$ that satisfies the following extended saddle-point condition (ESPC):

$$L_m\big((z^*, \alpha, \beta)^T\big) \leq L_m\big((z^*, \alpha^{**}, \beta^{**})^T\big) \leq L_m\big((z, \alpha^{**}, \beta^{**})^T\big) \tag{13}$$

for any $\alpha^{**} > \alpha^*$ and $\beta^{**} > \beta^*$ and for all $z \in \mathcal{N}_m(z^*)$, $\alpha \in \mathbb{R}^m$, and $\beta \in \mathbb{R}^r$.

Note that (13) can be satisfied under rather loose conditions because it is true for a range of penalty values and not for unique values. For this reason, we call $(z^*, \alpha^{**}, \beta^{**})^T$ an *extended saddle point* (ESP) of (12). The theorem leads to an easy way for finding CLM_m. Since an ESP is a local minimum of (12) (but not the converse), z^* can be found by gradually increasing the penalties of those violated constraints in (12) and by repeatedly finding the local minima of (12) until a feasible solution to P_m is obtained. The search for local minima can be accomplished by any existing local-search algorithm for unconstrained optimization.

Example 1 (cont'd). In solving (5), if we use $L_m((y, \alpha)^T) = f(y) + \alpha|y|$ and choose $\alpha^* = 1$ we have an ESP at $y^* = 0$ for any $\alpha^{**} > \alpha^*$. This establishes a local minimum of $L_m((y, \alpha)^T)$ at $y^* = 0$ with respect to $\mathcal{N}_d(y) = \{y - 1, y + 1\}$. Note that the α^* that satisfies Theorem 1 is only required to establish a local minimum of $L_m((y, \alpha)^T)$ at $y^* = 0$ and is, therefore, smaller than the α^* $(=\frac{4}{3})$ required to establish a global minimum of $L_m((y, \alpha)^T)$ in the static-penalty method. ∎

An important feature of the ESPC in Theorem 1 is that it can be partitioned in such a way that each subproblem implementing a partitioned condition can be solved by looking for any α^{**} and β^{**} that are larger than α^* and β^*.

Consider P_t, a version of P_m whose constraints can be partitioned into N subsets:

$$
\begin{array}{llllll}
(P_t): & \min\limits_{z} & f(z) & & & \\
& \text{subject to} & h^{(t)}(z(t)) = 0, & g^{(t)}(z(t)) \le 0 & \text{(local constraints)} & \\
& \text{and} & H(z) = 0, & G(z) \le 0 & \text{(global constraints).} &
\end{array}
\tag{14}
$$

Each subset of constraints can be treated as a subproblem, where Subproblem t, $t = 1, \ldots, N$, has local *state vector* $z(t) = (z_1(t), \ldots, z_{u_t}(t))^T$ of u_t mixed variables, and $\cup_{t=1}^{N} z(t) = z$. Here, $z(t)$ includes all the variables that appear in any of the m_t local equality constraint functions $h^{(t)} = \left(h_1^{(t)}, \ldots, h_{m_t}^{(t)}\right)^T$ and the r_t local inequality constraint functions $g^{(t)} = \left(g_1^{(t)}, \ldots, g_{r_t}^{(t)}\right)^T$. Since the partitioning is by constraints, $z(1), \ldots, z(N)$ may overlap with each other. Further, $z(g)$ includes all the variables that appear in any of the p global equality constraint functions $H = (H_1, \ldots, H_p)^T$ and the q global inequality constraint functions $G = (G_1, \ldots, G_q)^T$.

We first define N $_m(z)$, the mixed neighborhood of z for P_t, and decompose the ESPC in (13) into a set of necessary conditions that collectively are sufficient. Each partitioned ESPC is then satisfied by finding an ESP of the corresponding subproblem, and any violated global constraints are resolved by finding some appropriate penalties.

Definition 6. $\mathcal{N}_{p_0}(z)$, the *mixed neighborhood* of z for P_t when partitioned by its constraints, is:

$$
\mathcal{N}_{p_0}(z) = \bigcup_{t=1}^{N} \mathcal{N}_{p_1}^{(t)}(z) = \bigcup_{t=1}^{N} \left\{ z' \;\middle|\; z'(t) \in \mathcal{N}_{m_1}(z(t)) \text{ and } z'_i = z_i \;\forall z_i \notin z(t) \right\}, \tag{15}
$$

where $\mathcal{N}_{m_1}(z(t))$ is the mixed neighborhood of $z(t)$ (see Definition 2).

Intuitively, $\mathcal{N}_{m_1}(z(t))$ is separated into N neighborhoods, where the t^{th} neighborhood only perturbs the variables in $z(t)$ while leaving those variables in $z \backslash z(t)$ unchanged.

Without showing the details, we can consider P_t as a MINLP and apply Theorem 1 to derive its ESPC. We then decompose the ESPC into N necessary conditions, one for each subproblem, and an overall necessary condition on the global constraints across the subproblems. We first define the penalty function for Subproblem t.

Definition 7. Let $\Phi((z, \gamma, \eta)^T) = \gamma^T |H(z)| + \eta^T G(z)^+$ be the sum of the transformed global constraint functions weighted by their penalties, where $\gamma = (\gamma_1, \ldots, \gamma_p)^T \in \mathbb{R}^p$ and $\eta = (\eta_1, \ldots, \eta_q)^T$ are the penalty vectors for the global constraints. Then the penalty function for P_t in (14) and the corresponding penalty function in Subproblem t are defined as follows:

$$
L_m((z, \alpha, \beta, \gamma, \eta)^T) = f(z) + \sum_{t=1}^{N} \left\{ \alpha(t)^T |h^{(t)}(z(t))| + \beta(t)^T \left(g^{(t)}(z(t))\right)^+ \right\} + \Phi((z, \gamma, \eta)^T), \tag{16}
$$

$$\Gamma_m((z,\alpha(t),\beta(t),\gamma,\eta)^T) = f(z) + \alpha(t)^T|h^{(t)}(z(t))| + \beta(t)^T\left(g^{(t)}(z(t))\right)^+ + \Phi((z,\gamma,\eta)^T), \quad (17)$$

where $\alpha(t) = (\alpha_1(t),\ldots,\ \alpha_{m_t}(t))^T \in \mathbb{R}^{m_t}$ and $\beta(t) = (\beta_1(t),\ldots,\ \beta_{r_t}(t))^T \in \mathbb{R}^{r_t}$ are the penalty vectors for the local constraints in Subproblem *t*.

Theorem 2. *Partitioned necessary and sufficient ESPC on CLMm* of *Pt* [25]. Given $\mathcal{N}_{p_0}(z)$, the ESPC in (13) can be rewritten into *N* + 1 necessary conditions that, collectively, are sufficient:

$$\begin{aligned}\Gamma_m((z^*,\alpha(t),\beta(t),\gamma^{**},\eta^{**})^T) &\le \Gamma_m((z^*,\alpha(t)^{**},\beta(t)^{**},\gamma^{**},\eta^{**})^T)\\ &\le \Gamma_m((z,\alpha(t)^{**},\beta(t)^{**},\gamma^{**},\eta^{**})^T)\end{aligned} \quad (18)$$

$$L_m\left((z^*,\alpha^{**},\beta^{**},\gamma,\eta)^T\right) \le L_m\left((z^*,\alpha^{**},\beta^{**},\gamma^{**},\eta^{**})^T\right) \quad (19)$$

for any $\alpha(t)^{**} > \alpha(t)^* \ge 0,\ \beta(t)^{**} > \beta(t)^* \ge 0,\ \gamma^{**} > \gamma^* \ge 0,$ and $\eta^{**} > \eta^* \ge 0,$ and for all $z \in \mathcal{N}^{(t)}_{p_1}(z^*),\ \alpha(t) \in \mathbb{R}^{m_t},\ \beta(t) \in \mathbb{R}^{r_t},\ \gamma \in \mathbb{R}^p,\ \eta \in \mathbb{R}^q,$ and $t = 1,\ldots,N.$

Theorem 2 shows that the original ESPC in Theorem 1 can be partitioned into *N* necessary conditions in (18) and an overall necessary condition in (19) on the global constraints across the subproblems. Because finding an ESP to each partitioned condition is equivalent to solving a MINLP, we can reformulate the ESP search of the *t*th condition as the solution of the following optimization problem:

$$\left(P_t^{(t)}\right): \quad \begin{array}{cl}\min\limits_{z(t)} & f(z) + \gamma^T|H(z)| + \eta^T G(z)^+\\ \text{subject to} & h^{(t)}(z(t)) = 0 \quad \text{and} \quad g^{(t)}(z(t)) \le 0.\end{array} \quad (20)$$

The weighted sum of the global constraint functions in the objective of (20) is important because it leads to points that minimize the violations of the global constraints. When γ^T and η^T are large enough, solving $P_t^{(t)}$ will lead to points, if they exist, that satisfy the global constraints. Note that $P_t^{(t)}$ is very similar to the original problem and can be solved by the same solver to the original problem with some modifications on the objective function to be optimized.

In summary, we have shown in this section that the search for a CLM_m of P_m is equivalent to finding an ESP of the corresponding penalty function, and that this necessary and sufficient condition can be partitioned into multiple necessary conditions. The latter result allows the original problem to be decomposed by its constraints to multiple subproblems and to the reweighting of those violated global constraints defined by (19). The major benefit of this decomposition is that each subproblem involves only a fraction of the original constraints and is, therefore, a significant relaxation of the original problem with much lower complexity. The decomposition leads to a large reduction in the complexity of the original problem if the global constraints is small in quantity and can be resolved efficiently. We demonstrate in Section 5 that the number of global constraints in many benchmarks is indeed small when we exploit the locality of the constraints. In the next section, we describe our extensions to simulated annealing for finding ESPs.

3. Simulated annealing for constrained optimization

In this section, we present three algorithms for finding ESPs: the first two implementing the results in Theorems 1 and 2, and the third extending the penalty search algorithms in Section 2.1. All three methods are based on sampling the search space of a problem during their search and can be applied to solve continuous, discrete, and mixed-integer optimization problems. Without loss of generality, we only consider P_m with equality constraints, since an inequality constraint $g_j(z) \leq 0$ can be transformed into an equivalent equality constraint $g_j(z)^+ = 0$.

3.1 Constrained simulated annealing (CSA)

Figure 1 presents CSA, our algorithm for finding an ESP whose $(z^*, \alpha^{**})^T$ satisfies (13). In addition to probabilistic descents in the z subspace as in SA [18], with an acceptance probability governed by a temperature that is reduced by a properly chosen cooling schedule, CSA also does probabilistic ascents in the penalty subspace. The success of CSA lies in its strategy to search in the joint space, instead of applying SA to search in the subspace of the penalty function and updating the penalties in a separate phase of the algorithm. The latter approach would be taken in existing static and the dynamic penalty methods discussed in Section 2.1. CSA overcomes the limitations of existing penalty methods because it does not require a separate algorithm for choosing penalties. The rest of this section explains the steps of CSA [30, 28].

```
1.  procedure CSA
2.      set starting point $\mathbf{z} \leftarrow (z, \alpha)^T$ and initialize $\alpha \leftarrow 0$;
3.      set starting temperature $T \leftarrow T_0$ and cooling rate $0 < \kappa < 1$;
4.      set $N_T \leftarrow$ number of trials per temperature;
5.      while stopping condition is not satisfied do
6.          for $k \leftarrow 1$ to $N_T$ do
7.              generate trial point $\mathbf{z}' \in \mathcal{N}_m(\mathbf{z})$ using $G(\mathbf{z}, \mathbf{z}')$;
8.              if $\mathbf{z}'$ is accepted according to $A_T(\mathbf{z}, \mathbf{z}')$ then $\mathbf{z} \leftarrow \mathbf{z}'$
9.          end_for
10.         reduce temperature by $T \leftarrow \kappa T$;
11.     end_while
12. end_procedure
```

Figure 1. CSA: Constrained simulated annealing (see text for the initial values of the parameters). The differences between CSA and SA lie in their definitions of state **z**, neighborhood $\mathcal{N}_m(\mathbf{z})$, generation probability $G(\mathbf{z}, \mathbf{z}')$ and acceptance probability $A_T(\mathbf{z}, \mathbf{z}')$.

Line 2 sets a starting point $\mathbf{z} \leftarrow (z, \alpha)^T$, where z can be either user-provided or randomly generated (such as using a fixed seed 123 in our experiments), and α is initialized to zero. Line 3 initializes control parameter *temperature* T to be so large that almost any trial point $\mathbf{z}'$ will be accepted. In our experiments on continuous problems, we initialize T by first randomly generating 100 points of x and their corresponding neighbors

$x' \in \mathrm{N}_c(x)$ in close proximity, where $|x'_i - x_i| \leq 0.001$, and then setting $T = \max_{x,x',i}\left\{|L_m((x',1)^T) - L_m((x,1)^T)|, |h_i(x)|\right\}$. Hence, we use a large initial T if the function is rugged $(|L_m((x',1)^T) - L_m((x,1)^T)|$is large), or the function is not rugged but its constraint violation ($|h_i(x)|$) is large. We also initialize κ to 0.95 in our experiments.

Line 4 sets the number of iterations at each temperature. In our experiments, we choose $N_T \leftarrow \zeta\ (20n + m)$ where $\zeta \leftarrow 10(n + m)$, n is the number of variables, and m is the number of equality constraints. This setting is based on the heuristic rule in [9] using $n + m$ instead of n.

Line 5 stops CSA when the current **z** is not changed, *i.e.*, no other **z**′ is accepted, in two successive temperature changes, or when the current T is small enough (*e.g.* $T < 10^{-6}$).

Line 7 generates a random point $\mathbf{z}' \in \mathrm{N}_m(\mathbf{z})$ from the current $\mathbf{z} \in \mathcal{S} = \mathcal{Z} \times \Lambda$,, where $\Lambda = \mathbb{R}^m$ is the space of the penalty vector. In our implementation, $\mathrm{N}_m(\mathbf{z})$ consists of $(z', \alpha)_T$ and $(z, \alpha')_T$, where $z' \in \mathcal{N}_{m_1}(z)$ (see Definition 1), and $\alpha' \in \mathcal{N}_{m_2}(\alpha)$ is a point neighboring to α when $h(z) \neq 0$:

$$\mathcal{N}_m(\mathbf{z}) \;=\; \left\{(z', \alpha)^T \in \mathcal{S} \text{ where } z' \in \mathcal{N}_{m_1}(z)\right\} \cup \left\{(z, \alpha')^T \in \mathcal{S} \text{ where } \alpha' \in \mathcal{N}_{m_2}(\alpha)\right\} \tag{21}$$

and $$\mathcal{N}_{m_2}(\alpha) \;=\; \left\{\alpha' \in \Lambda \text{ where } (\alpha'_i < \alpha_i \text{ or } \alpha'_i > \alpha_i \text{ if } h_i(z) \neq 0) \text{ and } (\alpha'_i = \alpha_i \text{ if } h_i(z) = 0)\right\}. \tag{22}$$

According to this definition, α_i is not perturbed when $h_i(z) = 0$ is satisfied.

$G(\mathbf{z}, \mathbf{z}')$, the *generation probability* from **z** to $\mathbf{z}' \in \mathrm{N}_m(\mathbf{z})$, satisfies:

$$0 \geq G(\mathbf{z}, \mathbf{z}') \leq 1 \quad \text{and} \quad \sum_{\mathbf{z}' \in \mathcal{N}_m(\mathbf{z})} G(\mathbf{z}, \mathbf{z}') = 1. \tag{23}$$

Since the choice of $G(\mathbf{z}, \mathbf{z}')$ is arbitrary as long as it satisfies (23), we select **z**′ in our experiments with uniform probability across all the points in $\mathrm{N}_m(\mathbf{z})$, independent of T:

$$G(\mathbf{z}, \mathbf{z}') \;=\; \frac{1}{|\mathcal{N}_m(\mathbf{z})|}. \tag{24}$$

As we perturb either z or α but not both simultaneously, (24) means that **z**′ is generated either by choosing $z' \in \mathcal{N}_{m_1}(z)$ randomly or by generating α' uniformly in a predefined range.

Line 8 accepts **z**′ with acceptance probability $A_T(\mathbf{z}, \mathbf{z}')$ that consists of two components, depending on whether z or α is changed in **z**′:

$$A_T(\mathbf{z}, \mathbf{z}') = \begin{cases} \exp\left(-\frac{(L_m(\mathbf{z}') - L_m(\mathbf{z}))^+}{T}\right) & \text{if } \mathbf{z}' = (z', \alpha)^T \\ \exp\left(-\frac{(L_m(\mathbf{z}) - L_m(\mathbf{z}'))^+}{T}\right) & \text{if } \mathbf{z}' = (z, \alpha')^T. \end{cases} \tag{25}$$

The acceptance probability in (25) differs from the acceptance probability used in conventional SA, which only has the first case in (25) and whose goal is to look for a global minimum in the z subspace. Without the α subspace, only probabilistic descents in the z subspace are carried out.

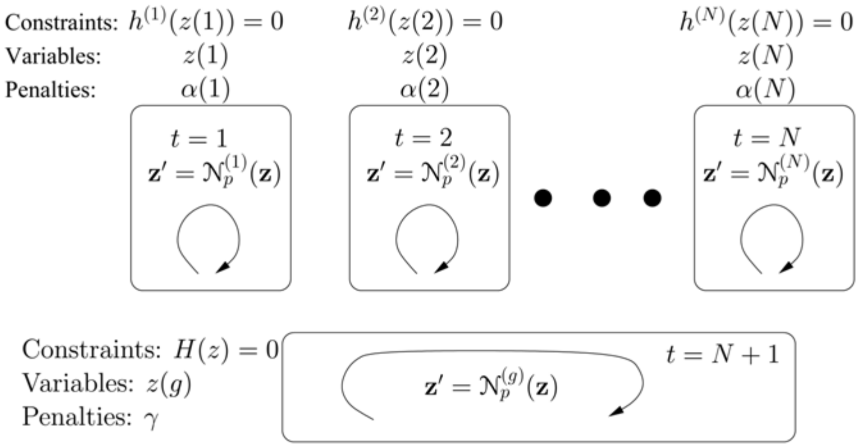
Constraints: $h^{(1)}(z(1)) = 0$
$h^{(2)}(z(2)) = 0$
$h^{(N)}(z(N)) = 0$
Variables: $z(1)$
$z(2)$
$z(N)$
Penalties: $\alpha(1)$
$\alpha(2)$
$\alpha(N)$
$t = 1$
$\mathbf{z}' = \mathcal{N}_p^{(1)}(\mathbf{z})$
$t = 2$
$\mathbf{z}' = \mathcal{N}_p^{(2)}(\mathbf{z})$
$t = N$
$\mathbf{z}' = \mathcal{N}_p^{(N)}(\mathbf{z})$
Constraints: $H(z) = 0$
Variables: $z(g)$
Penalties: γ
$t = N + 1$
$\mathbf{z}' = \mathcal{N}_p^{(g)}(\mathbf{z})$

Figure 2. CPSA: Constraint-partitioned simulated annealing.

In contrast, our goal is to look for an ESP in the joint $Z \times \Lambda$ space, each existing at a local minimum in the z subspace and at a local maximum in the α subspace. To this end, CSA carries out *probabilistic descents* of $L_m\left((z,\alpha)^T\right)$ with respect to z for each fixed α. That is, when we generate a new z' under a fixed α, we accept it with probability one when $\delta_z = L_m\left((z',\alpha)^T\right) - L_m\left((z,\alpha)^T\right)$ is negative; otherwise, we accept it with probability $e^{-\delta_z/T}$. This step has exactly the same effect as in conventional SA; that is, it performs descents with occasional ascents in the z subspace.

However, descents in the z subspace alone will lead to a local/global minimum of the penalty function without satisfying the corresponding constraints. In order to satisfy all the constraints, CSA also carries out *probabilistic ascents* of $L_m\left((z,\alpha)^T\right)$ with respect to α for each fixed z in order to increase the penalties of violated constraints and to force them into satisfaction. Hence, when we generate a new α' under a fixed z, we accept it with probability one when $\delta_\alpha = L_m\left((z,\alpha')^T\right) - L_m\left((z,\alpha)^T\right)$ is positive; otherwise, we accept it with probability $e^{-\delta_\alpha/T}$. This step is the same as that in conventional SA when performing ascents with occasional descents in the α subspace. Note that when a constraint is satisfied, the corresponding penalty will not be changed according to (22).

Finally, Line 10 reduces T by the following *cooling schedule* after looping N_T times at given T:

$$T \longleftarrow \kappa \cdot T \quad \text{where the cooling-rate constant } \kappa \leftarrow 0.95 \text{ (typically } 0.8 \leq \kappa \leq 0.99). \tag{26}$$

At high T, (25) allows any trial point to be accepted with high probabilities, thereby allowing the search to traverse a large space and overcome infeasible regions. When T is reduced, the acceptance probability decreases, and at very low temperatures, the algorithm behaves like a local search.

3.2 Constraint-Partitioned Simulated Annealing (CPSA)

We present in this section CPSA, an extension of CSA that decomposes the search in CSA into multiple subproblems after partitioning the constraints into subsets. Recall that, according to Theorem 2, P_t in (14) can be partitioned into a sequence of N subproblems defined in (20) and an overall necessary condition defined in (19) on the global constraints across the subproblems, after choosing an appropriate mixed neighborhood. Instead of considering all the constraints together as in CSA, CPSA performs searches in multiple subproblems, each involving a small subset of the constraints. As in CSA, we only consider P_t with equality constraints.

Figure 2 illustrates the idea in CPSA. Unlike the original CSA that solves the problem as a whole, CPSA solves each subproblem independently. In Subproblem t, $t = 1, \ldots, N$, CSA is performed in the $(z(t), \alpha(t))^T$ subspace related to the local constraints $h^{(t)}(z(t)) = 0$. In addition, there is a global search that explores in the $(z(g), \gamma)^T$ subspace on the global constraints $H(z) = 0$. This additional search is needed for resolving any violated global constraints.

```
1. procedure CPSA
2.     set starting point $\mathbf{z} \leftarrow (z, \alpha, \gamma)^T$ and initialize $\alpha = \gamma \leftarrow 0$;
3.     set starting temperature $T \leftarrow T^0$ and cooling rate $0 < \kappa < 1$;
4.     set $N_T \leftarrow$ number of trials per temperature;
5.     while stopping condition is not satisfied do
6.         for $k \leftarrow 1$ to $N_T$ do
7.             set $t$ to be a random integer between 1 and $N + 1$;
8.             if $1 \le t \le N$ then
9.                 generate $\mathbf{z}' \in \mathcal{N}_p^{(t)}(\mathbf{z})$ using $G^{(t)}(\mathbf{z}, \mathbf{z}')$;
10.                if $\mathbf{z}'$ is accepted according to $A_T(\mathbf{z}, \mathbf{z}')$ then $\mathbf{z} \leftarrow \mathbf{z}'$;
11.            else /* $t = N + 1$ */
12.                generate $\mathbf{z}' \in \mathcal{N}_p^{(g)}(\mathbf{z})$ using $G^{(g)}(\mathbf{z}, \mathbf{z}')$;
13.                if $\mathbf{z}'$ is accepted according to $A_T(\mathbf{z}, \mathbf{z}')$ then $\mathbf{z} \leftarrow \mathbf{z}'$;
14.            end_if
15.        end_for
16.        reduce temperature by $T \leftarrow \kappa T$;
17.    end_while
18. end_procedure
```

Figure 3. The CPSA search procedure.

Figure 3 describes the CPSA procedure. The first six lines are similar to those in CSA.

To facilitate the convergence analysis of CPSA in a Markov-chain model, Lines 7-14 randomly pick a subproblem for evaluation, instead of deterministically enumerating the subproblems in a round-robin fashion, and stochastically accept a new probe using an acceptance probability governed by a decreasing temperature. This approach leads to a memoryless Markovian process in CPSA.

Line 7 randomly selects Subproblem i, $i = 1 \ldots, N+1$, with probability $P_s(t)$, where $P_s(t)$ can be arbitrarily chosen as long as:

$$\sum_{t=1}^{N+1} P_s(t) = 1 \text{ and } P_s(t) > 0. \tag{27}$$

When t is between 1 and N (Line 8), it represents a local exploration step in Subproblem t. In this case, Line 9 generates a trial point $\mathbf{z}' \in \mathcal{N}_p^{(t)}(\mathbf{z})$ from the current point $\mathbf{z} = (z, \alpha, \gamma)^T \in \mathcal{S}$ using a *generation probability* $G^{(t)}(\mathbf{z}, \mathbf{z}')$ that can be arbitrary as long as the following is satisfied:

$$0 \le G^{(t)}(\mathbf{z}, \mathbf{z}') \le 1 \quad \text{and} \quad \sum_{\mathbf{z}' \in \mathcal{N}_p^{(t)}(\mathbf{z})} G^{(t)}(\mathbf{z}, \mathbf{z}') = 1. \tag{28}$$

The point is generated by perturbing $z(t)$ and $\alpha(t)$ in their neighborhood $\mathcal{N}_p^{(t)}(\mathbf{z})$:

$$\mathcal{N}_p^{(t)}(\mathbf{z}) = \left\{(z', \alpha(t), \gamma) \in \mathcal{S} \mid z' \in \mathcal{N}_{p_1}^{(t)}(z)\right\} \cup \left\{(z, \alpha'(t), \gamma) \in \mathcal{S} \mid \alpha'(t) \in \mathcal{N}_{p_2}^{(t)}(\alpha(t))\right\} \tag{29}$$

$$\begin{aligned} \mathcal{N}_{p_2}^{(t)}(\alpha(t)) = &\left\{\alpha'(t) \in \Lambda^{(\alpha(t))} \text{ where } (\alpha_i'(t) < \alpha_i(t) \text{ or } \alpha_i'(t) > \alpha_i(t) \text{ if } h_i(z(t)) \ne 0)\right. \\ &\left.\text{and } (\alpha_i'(t) = \alpha_i(t) \text{ if } h_i(z(t)) = 0)\right\}, \end{aligned} \tag{30}$$

and $\mathcal{N}_{p_1}^{(t)}(z)$ is defined in (15) and $\Lambda^{(\alpha(t))} = \mathbb{R}^{m_t}$. This means that $\mathbf{z}' \in \mathcal{N}_p^{(t)}(\mathbf{z})$ only differs from $\mathbf{z}$ in $z(t)$ or $\alpha(t)$ and remains the same for the other variables. This is different from CSA that perturbs $\mathbf{z}$ in the overall variable space. As in CSA, α_i is not perturbed when $h_i(z(t)) = 0$ is satisfied. Last, Line 10 accepts $\mathbf{z}'$ with the Metropolis probability $A_T(\mathbf{z}, \mathbf{z}')$ similar to that in (25):

$$A_T(\mathbf{z}, \mathbf{z}') = \begin{cases} \exp\left(-\frac{(L_m(\mathbf{z}') - L_m(\mathbf{z}))^+}{T}\right) & \text{if } \mathbf{z}' = (z', \alpha, \gamma)^T \\ \exp\left(-\frac{(L_m(\mathbf{z}) - L_m(\mathbf{z}'))^+}{T}\right) & \text{if } \mathbf{z}' = (z, \alpha', \gamma)^T \text{ or } \mathbf{z}' = (z, \alpha, \gamma')^T. \end{cases} \tag{31}$$

When $t = N + 1$ (Line 11), it represents a global exploration step. In this case, Line 12 generates a random trial point $\mathbf{z}' \in \mathcal{N}_p^{(g)}(\mathbf{z})$ using a generation probability $G^{(g)}(\mathbf{z}, \mathbf{z}')$ that satisfies the condition similar to that in (28). Assuming $\mathcal{N}_{m_1}(z(g))$ to be the mixed neighborhood of $z(g)$ and $\Lambda^{(g)} = \mathbb{R}^p$, $\mathbf{z}'$ is obtained by perturbing $z(g)$ and γ in their neighborhood $\mathcal{N}_p^{(g)}(\mathbf{z})$::

$$\mathcal{N}_p^{(g)}(\mathbf{z}) = \left\{(z', \alpha, \gamma)^T \in \mathcal{S} \text{ where } z' \in \mathcal{N}_{p_1}^{(g)}(z)\right\} \cup \left\{(z, \alpha, \gamma')^T \in \mathcal{S} \text{ where } \gamma' \in \mathcal{N}_{p_2}^{(g)}(\gamma)\right\} \tag{32}$$

$$\mathcal{N}_{p_1}^{(g)}(z) = \left\{z' \text{ where } z'(g) \in \mathcal{N}_{m_1}(z(g)) \text{ and } z_i' = z_i \ \forall z_i \notin z(g)\right\} \tag{33}$$

$$\mathcal{N}_{p_2}^{(g)}(\gamma) \;=\; \{\gamma' \in \Lambda^{(g)} \text{ where } (\gamma_i' < \gamma_i \text{ or } \gamma_i' > \gamma_i \text{ if } H_i(z) \neq 0) \text{ and } (\gamma_i' = \gamma_i \text{ if } H_i(z) = 0)\}. \quad (34)$$

Again, $\mathbf{z}'$ is accepted with probability $A_T(\mathbf{z}, \mathbf{z}')$ in (31) (Line 13). Note that both $\mathcal{N}_p^{(t)}(\mathbf{z})$ ($\mathbf{z}$) and $\mathcal{N}_p^{(g)}(\mathbf{z})$: ensure the ergodicity of the Markov chain, which is required for achieving asymptotic convergence.

When compared to CSA, CPSA reduces the search complexity through constraint partitioning. Since both CSA and CPSA need to converge to an equilibrium distribution of variables at a given temperature before the temperature is reduced, the total search time depends on the convergence time at each temperature. By partitioning the constraints into subsets, each subproblem only involves an exponentially smaller subspace with a small number of variables and penalties. Thus, each subproblem takes significantly less time to converge to an equilibrium state at a given temperature, and the total time for all the subproblems to converge is also significantly reduced. This reduction in complexity is experimentally validated in Section 5.

3.3 Greedy ESPC Search Method (GEM)

In this section, we present a dynamic penalty method based on a greedy search of an ESP. Instead of probabilistically accepting a probe as in CSA and CPSA, our greedy approach accepts the probe if it improves the value of the penalty function and rejects it otherwise.

One simple approach that does not work well is to gradually increase α^{**} until $\alpha^{**} > \alpha^*$, while minimizing the penalty function with respect to z using an existing local-search method. This simple iterative search does not always work well because the penalty function has many local minima that satisfy the second inequality in (13), but some of these local minima do not satisfy the first inequality in (13) even when $\alpha^{**} > \alpha^*$. Hence, the search may generate stationary points that are local minima of the penalty function but are not feasible solutions to the original problem.

To address this issue, Figure 4 shows a global search called the *Greedy ESPC Search Method* [32] (GEM). GEM uses the following penalty function:

$$L_g\big((z,\alpha)^T\big) = f(z) + \sum_{i=1}^{m} \alpha_i |h_i(z)| + \frac{1}{2}||h(z)||^2. \quad (35)$$

Lines 5-8 carries out Ng iterative descents in the z subspace. In each iteration, Line 6 generates a probe $z' \in \mathcal{N}_{m_1}(z)$ neighboring to z. As defined in (24) for CSA, we select z' with uniform probability across all the points in $\mathcal{N}_{m_1}(z)$. Line 7 then evaluates $L_g\ ((z', \alpha)^T)$ and accepts z' only when it reduces the value of L_g. After the N_g descents, Line 9 updates the penalty vector α in order to bias the search towards resolving those violated constraints.

When α^{**} reaches its upper bound during a search but a local minimum of L_g does not correspond to a CLM_m of P_m, we can reduce α^{**} instead of restarting the search from a new starting point. The decrease will change the terrain of L_g and "lower" its barrier, thereby allowing a local search to continue in the same trajectory and move to another local minimum of L_g. In Line 10, we reduce the penalty value of a constraint when its maximum violation is not reduced for three consecutive iterations. To reduce the penalties, Line 11 multiplies each element in α by a random real number uniformly generated between 0.4 to 0.6. By repeatedly increasing α^{**} to its upper bound and by reducing it to some lower bound, a local search will be able to escape from local traps and visit multiple local minima of the penalty function. We leave the presentation of the parameters used in GEM and its experimental results to Section 5.

```
1. procedure GEM
2.     set $\varrho$ to be a positive real constant;
3.     set starting point $\mathbf{z} \leftarrow (z, \alpha)^T$ and initialize $\alpha$;
4.     repeat
5.         for k $\leftarrow$ 1 to $N_g$ /* $N_g \leftarrow 20$, a positive integer in our experiments */
6.             generate random trial point $z' \in \mathcal{N}_{m_1}(z)$;
7.             if $(L_g((z,\alpha)^T) > L_g((z',\alpha)^T))$ then $z' \leftarrow z$; end_if
8.         end_for
9.         update $\alpha \longleftarrow \alpha + \varrho|h(z)|$;
10.        if (condition to decrease $\alpha$ is satisfied) then
11.            reduce $\alpha$ in order to allow the search to escape from local traps;
12.        end_if
13.    until stopping conditions are satisfied;
14. end_procedure
```

Figure 4. Greedy ESPC search method (GEM).

4. Asymptotic convergence of CSA and CPSA

In this subsection, we show the asymptotic convergence of CSA and CPSA to a constrained global minimum in *discrete* constrained optimization problems. Without repeating the definitions in Section 1, we can similarly define a discrete nonlinear programming problem (P_d), a discrete neighborhood ($\mathcal{N}_d(y)$), a discrete constrained local minimum (CLM_d), a discrete constrained global minimum (CGM_d), and a penalty function in discrete space (L_d).

4.1 Asymptotic convergence of CSA

We first define the asymptotic convergence property. For a global minimization problem, let Ω be its search space, Ω_s be the set of all global minima, and $\omega(j) \in \Omega$, $j = 0, 1, \ldots$, be a sequence of points generated by an iterative procedure ψ until some stopping conditions hold.

Definition 8. Procedure ψ is said to have *asymptotic convergence to a global minimum*, or simply *asymptotic convergence* [2], if ψ converges with probability one to an element in Ω_s; that is, $\lim_{j\to\infty} P(\omega(j) \in \Omega_s) = 1$, independent of $\omega(0)$, where $P(w)$ is the probability of event w.

In the following, we first state the result on the asymptotic convergence of CSA to a CGM_d of P_d with probability one when T approaches 0 and when T is reduced according to a specific cooling schedule. By modeling CSA by an inhomogeneous Markov chain, we show that the