Journal of Global Optimization, , 1–36 ()

Trace-Based Methods for Solving Nonlinear Global Optimization and Satisfiability Problems *

BENJAMIN W. WAH b-wah@uiuc.edu
Department of Electrical and Computer Engineering and the Coordinated Science Laboratory, University of Illinois, Urbana-Champaign, Urbana, IL 61801, USA

YAO-JEN CHANG chang@wavelet.cycu.edu.tw
Department of Electronic Engineering, Chung Yuan Christian University, Chung-Li, Taiwan

Received July 18, 1995

Editor: Jun Gu

Abstract. In this paper we present a method called *NOVEL* (Nonlinear Optimization via External Lead) for solving continuous and discrete global optimization problems. NOVEL addresses the balance between global search and local search, using a trace to aid in identifying promising regions before committing to local searches. We discuss NOVEL for solving continuous constrained optimization problems and show how it can be extended to solve constrained satisfaction and discrete satisfiability problems. We first transform the problem using Lagrange multipliers into an unconstrained version. Since a stable solution in a Lagrangian formulation only guarantees a local optimum satisfying the constraints, we propose a global search phase in which an aperiodic and bounded trace function is added to the search to first identify promising regions for local search. The trace generates an information-bearing trajectory from which good starting points are identified for further local searches. Taking only a small portion of the total search time, this elegant approach significantly reduces unnecessary local searches in regions leading to the same local optimum. We demonstrate the effectiveness of NOVEL on a collection of continuous optimization benchmark problems, finding the same or better solutions while satisfying the constraints. We extend NOVEL to discrete constraint satisfaction problems (CSPs) by showing an efficient transformation method for CSPs and the associated representation in finite-difference equations in NOVEL. We apply NOVEL to solve Boolean satisfiability instances in circuit fault detection and circuit synthesis applications, and show comparable performance when compared to the best existing method.

Keywords: Augmented Lagrange multiplier method, continuous nonlinear programming problems, constraint satisfaction problems, satisfiability problems, trace function, trajectory-based method.

1. Introduction

In this paper we consider constrained global optimization problems over continuous and discrete variables. For continuous problems, we show an elegant global search method that traverses the search space in a continuous and effective manner. We show extensions of our proposed method to constraint satisfaction problems (CSPs)

* Research was supported in part by National Science Foundation Grants MIP 92-18715 and MIP 96-32316 and in part by Joint Services Electronics Program Contract N00014-90-J-1270.

and discrete problems. In the latter case, we use satisfiability problems (SATs) to demonstrate the effectiveness of our proposed method.

Global minimization looks for the minimizer x that is no larger than any other local minimum x^* [45, 23, 94, 63], whereas local minimization aims at finding a local minimum x^*. Finding the global optimum x^{**} is a challenging problem as there may not be enough time to find a feasible solution, and even when a feasible solution is found, we have no way of showing that it is optimal. As stated by Griewank [28], global optimization is mathematically ill-posed in the sense that a lower bound for $f(x)$ cannot be given after any finite number of evaluations, unless f satisfies certain subsidiary conditions such as Lipschitz condition and the condition that the search area is bounded. Standard nonlinear programming methods usually obtain a local minimum or a stationary point satisfying the constraints. Such a local solution is global only when $f(x)$ is quasi-convex and the feasible region is convex, which rarely happens in practice [23].

Global optimization is a complex process comprised of searching different regions of attraction and balancing the computation between global search and local refinement. In an optimization problem, a region of attraction defines the region inside, or on the rim of, which there is a minimum and the constraints are satisfied. The rim of a region is a *divide* that separates it from others. Due to nonlinearity, global optimization is often performed without *a priori* knowledge of problem terrains or regions of attraction [94, 97, 72, 58]. Therefore, global optimization algorithms use heuristic global measures to search for new regions of attraction at run time. Promising regions identified are further optimized by local refinement procedures, such as gradient descent and Newton's methods.

In this paper, we propose a new method, called *NOVEL* (*Nonlinear Optimization via External Lead*), for solving constrained global optimization problems. Our algorithm can be viewed as a dynamic system that takes as input a trace and generates a trajectory to collect terrain information during the global-exploration process. The trace is a user-designed continuous aperiodic curve that advances with time. When combined with local gradients, the continuous trace evolves into an information-bearing trajectory that finds new regions of attraction. Based on the trajectory, promising starting points are identified from which existing local optimization methods are applied to find exact local minima.

We have applied *NOVEL* to two classes of application problems in this paper. The first class is a collection of constrained global optimization benchmark problems [23] derived from a variety of engineering applications. Unlike small artificial benchmark problems [3, 90, 5, 20], most of these problems are non-convex and have sizes ranging from small (tens of variables) to medium to large (hundreds of variables). Many of them have their best known solutions reported by others in the literature; however, the optimal solutions are generally unknown. As a result, they represent a challenging class of problems to be studied by any optimization algorithm. In Section 5 we show improved solutions to some of the problems summarized in Table 1.

Table 1. Summary of constrained continuous benchmark problems reported in [23] on some engineering applications and the ranges of the number of variables and the number of constraints in each problem.

Application Problem	No. Variables	No. Constraints
Pooling/Blending	[5,10]	[5,10]
VLSI Compaction Design	$[10^2,10^5]$	$[10^3,10^6]$
Pressure Vessel Design	[15,20]	[40,50]
Distillation Column Sequencing	[30,90]	[30,70]
Reactor-Separator-Recycle System	[100-120]	[80,100]
Complex Chemical Reactor Network	[40,110]	[30,100]
Heat Exchanger Network Synthesis	[10,60]	[10,40]
Speed Reducer Weight Minimization	[5,10]	[10,20]
Phase and Chemical Reaction Equilibrium	[7,10]	[4,13]

The second class of application problems we have studied are the satisfiability (SAT) problem described in the DIMACS benchmark suite. *NOVEL*, as a general method for global optimization, shows competitive results when compared to the best existing methods designed for these problems.

This paper is organized as follows. Section 2 formulates the problems for global optimization. Previous works on global optimization are summarized in Section 3. We then describe *NOVEL* in Section 4. Experimental results on nonlinear constrained optimization are reported in Section 5, and those for SAT are in Section 6. Concluding remarks are drawn in Section 7.

2. Problem Formulations and Transformations

In this section we show a unified formulation of global search problems. These problems can be grouped into three classes.

- *Continuous optimization problems with objectives.* These are constrained nonlinear optimization problems over continuous variables and are the basic form solved by our global optimization algorithm. Each has a (nonlinear) objective function and a set of possibly nonlinear constraints.

- *Continuous problems without objectives.* These are also called CSPs or feasibility problems, as the goal is to find feasible solutions that satisfy the constraints.

- *Discrete problems, with or without objective.* We study SAT, a special problem in this class.

In the remaining sections, we only study in detail problems in the first class and SAT problems in the last class. For the second class, we show its formulation in this section, and will show its evaluation in a future paper.

2.1. Constrained Nonlinear Global Optimization Problems

They take the following form.

$$\begin{aligned} &\text{minimize} \quad f(x) \\ &\text{subject to} \quad h(x) = 0 \\ &\qquad\qquad g(x) \leq 0 \end{aligned} \tag{1}$$

where $x \in R^n$, $f(x)$ is an objective function, $h(x)$ represents a set of m equality constraints, and $g(x)$, a set of q inequality constraints. All $f(x)$, $h(x)$, and $g(x)$ are assumed to be differentiable.

2.2. Continuous Constraint Satisfaction Problems (CSPs)

These are solved when a satisfying assignment of variables is found. An equality-constrained CSP can be defined as follows.

$$\text{Find a feasible solution that satisfies} \quad h(x) = 0 \tag{2}$$

A CSP can be transformed into a constrained optimization problem by adding a merit function that measures the norm of the constraint set.

$$\begin{aligned} &\text{minimize} \quad N(h(x)) \\ &\text{subject to} \quad h(x) = 0 \end{aligned} \tag{3}$$

where $x \in R^n$, $h(x)$ is defined so that it is differentiable, and $N(.)$ is a scalar differentiable function that takes the norm of its argument so that $N(h(x)) = 0$ iff $h(x) = 0$. Although (3) is equivalent to the original constraint-only formulation (2), the objective (merit) function indicates how close the constraints are being satisfied, hence providing additional guidance in leading to a satisfiable assignment.

Note that a sufficient condition for solving the CSP defined in (2) is when there is an assignment such that the objective function in (3) is zero. However, optimizing the objective function in (3) alone without the constraints is less effective as there may exist many local minima in the space of $N(.)$. Strictly following a descent path of $N(.)$ often ends up in a dent where $N(.)$ is not 0 but its value cannot be improved by local refinement.

2.3. Discrete CSPs

These are similar to continuous CSPs except that their variables take only discrete values. For an equality-constrained discrete CSP, $N(h(x))$, the norm in (3), can be defined as the number of conflicts in $h(x)$, where $x \in Z$ for general integer CSPs and $x \in \{0, 1\}$ for binary CSPs. There are various ways to define conflicts; an example of which is the number of unsatisfied constraints. The formulation is stated as follows.

$$\begin{aligned} &\text{minimize} \quad N(h(x)) \\ &\text{subject to} \quad h(x) = 0, \quad \text{where } x \in Z \end{aligned} \tag{4}$$

Many combinatorial problems, such as scheduling, coloring and Boolean satisfiability can be formulated as discrete CSPs.

Satisfiability (SAT) is an important class of discrete CSPs that we study in this paper. It can be used to model many problems in artificial intelligence, computer aided design, database query, and planning, just to name a few. They are known to be NP-complete and require algorithms of exponential complexity in the worst case in order to obtain a satisfying assignment.

A SAT problem is defined as follows. Given a set of clauses $C_1, C_2, \cdots, C_n$ on variables $x_1, x_2, \cdots, x_m$ and a Boolean formula in a conjunctive normal form (CNF)

$$C_1 \cap C_2 \cap \cdots \cap C_n, \tag{5}$$

find an assignment of values to the variables so that (5) evaluates to *true*, or derive its infeasibility if the problem is infeasible.

Instead of solving (5) directly, the problem can be reformulated as a global optimization problem in which the goal is to minimize $N(x)$, the number of unsatisfiable clauses. That is,

$$\text{minimize} \quad N(x) = \sum_{i=1}^{n} U_i(x) \tag{6}$$

where $U_i(x)$ equals 0 if the logical assignment x satisfies C_i and 1 otherwise. In this case, $N(x)$ equals 0 when all the clauses are satisfied.

Unfortunately, $N(x)$ defined in (6) has many dent-like local optima [56, 75, 76, 33, 34, 35], where a local minimum is a state where its local neighborhood does not include any state that is strictly better. Consequently, descent or hill-climbing methods can get trapped at local minima, and restarts merely bring the search to another local minimum.

A better way to handle SAT problems is to formulate them as constrained optimization problems. The formulation we have adopted in this paper is as follows.

$$\begin{aligned} &\text{minimize} \quad N(x) = \sum_{i=1}^{n} U_i(x) \\ &\text{subject to} \quad U_i(x) = 0 \quad \forall i \in \{1, 2, \ldots, n\} \end{aligned} \tag{7}$$

This formulation overcomes the deficiency of (6) because when the search is trapped in a local minimum, the unsatisfied clauses defined in the constraints can be used to provide a force to bring the search out of the local minimum.

2.4. Handling Constraints

Transformational and *non-transformational* techniques are two important classes of techniques developed to solve constrained local optimization problems. They can be adopted to handle constraints in constrained global optimization.

Non-transformational approaches include discarding and back-to-feasible-regions methods. The former [47, 55] drop solutions once they were found to be infeasible, and the latter [48] attempt to maintain feasibility by reflecting moves from boundaries if such moves go off the current feasible region. Both methods have been combined with global search and do not involve transformation to relax constraints.

Transformational approaches, on the other hand, convert the original problem into another form before solving them. Well known methods include penalty, barrier, and Lagrange-multiplier methods [54]. Penalty methods transform constraints into part of the objective function and require tuning penalty coefficients either before or during the run. Barrier methods are similar except that barriers are set up to avoid solutions from going out of feasible regions. Penalty and barrier methods are inexact methods that cannot guarantee accurate optimal solutions at the end, although they can be combined with other methods to overcome the accuracy problem.

The Lagrange-multiplier method is an important means of managing numerical stability and achieving solution accuracy at a price of increased number of problem dimensions. By introducing Lagrange multipliers, constraints can be gradually resolved through iterative updates. Lagrange and augmented Lagrange methods are exact methods that optimize the objective with great precision by attempting to meet the Kuhn-Tucker conditions. In view of their advantages, we use Lagrange multipliers for constraint relaxation in developing our algorithm.

Using Lagrange multipliers, we have reformulated continuous CSPs (3) and discrete CSPs (4) into constrained global optimization problems. These are shown in the next two subsections.

2.5. Lagrange Transformation for Continuous Problems

We handle constraints in continuous problems by Lagrange transformation. The transformed problems are used in Section 4 when we present our global optimization algorithm.

In general, constrained optimization problems may include equality and inequality constraints. In this paper, we first transform an inequality constraint into an equality constraint by adding a slack variable and by introducing a new inequality constraint on the slack variable. Other better approaches, such as that discussed by Luenberger [54], have been studied in our recent work but are not reported here. The optimization problem with equality constraints is shown as follows.

$$\begin{aligned} &\text{minimize} \quad f(x) \\ &\text{subject to} \quad h(x) = 0 \end{aligned} \tag{8}$$

The **Lagrangian function** of (8) can be represented as follows.

$$L(x, \lambda) = f(x) + \lambda^T h(x) \tag{9}$$

where λ is the corresponding set of Lagrange multipliers. According to classic optimization theory, all the extrema of (8), whether local or global, are roots of the

following set of equations.

$$\nabla_x L(x,\lambda) = 0 \tag{10}$$
$$\nabla_\lambda L(x,\lambda) = 0 \tag{11}$$

These two equations form the sufficient conditions to guarantee the optimality to the solution of (8).

To provide better numerical stability in convergence, we can use instead the **augmented Lagrangian formulation**. The augmented Lagrangian of the optimization problem (8) is defined as follows.

$$\mathcal{L}(x,\lambda) = f(x) + ||h(x)||_2^2 + \lambda^T h(x) \tag{12}$$

Again, our task is to find zeros for

$$\nabla_x \mathcal{L}(x,\lambda) = 0 \tag{13}$$
$$\nabla_\lambda \mathcal{L}(x,\lambda) = 0 \tag{14}$$

To solve (13) and (14), we form a Lagrangian dynamic system that includes a set of dynamic equations to seek equilibrium points along a gradient path. These equilibrium points are saddle-points of (13) and (14), which correspond to constrained minima of the underlying optimization problem (8). The Lagrangian dynamic system can be described as follows.

$$\frac{dx}{dt} = -\nabla_x \mathcal{L}(x,\lambda) \tag{15}$$
$$\frac{d\lambda}{dt} = \nabla_\lambda \mathcal{L}(x,\lambda) \tag{16}$$

Note that when (13) and (14) are solved at the same time, equilibrium holds when both right-hand sides are zero. Further, note that equilibrium is a saddle point because (15) has a minus sign that optimizes the Lagrangian function in the space of the original variables along a negative gradient path, whereas (16) optimizes the function in the space of the Lagrange multipliers along an ascending path.

Methods for solving (15) and (16) are based on local search methods. When applied, an initial assignment to x and λ are first given, and the local solution will be the very saddle point reached from this initial point. After reaching the saddle point, the solution will not improve unless a new starting point is selected. Note that nonlinearity can cause chaos in which a small variation in the initial point can lead to a completely different solution. This happens when (15) and (16) are implemented using a finite-step method that uses a line search to progress along a gradient-like direction in each iteration. In this case, over-shoots and under-shoots can lead to unpredictable and perhaps undesirable solutions.

The Lagrange method can be used to solve CSPs formulated in (3). Define the Lagrangian for (3) as

$$L(x,\lambda) = N(h(x)) + \lambda^T h(x) \tag{17}$$

The corresponding differential equations used to seek saddle points are as follows.

$$\frac{dx}{dt} = -\bigtriangledown_x L(x,\lambda) = -\bigtriangledown_x N(h(x)) - \lambda^T \bigtriangledown_x h(x) \quad (18)$$

$$\frac{d\lambda}{dt} = \bigtriangledown_\lambda L(x,\lambda) = h(x) \quad (19)$$

Note that any local minimum in the Lagrangian space is also the global minimum. Hence, the search will stop when a satisfiable solution is reached. This is unlike the case when only the original x variables are optimized.

2.6. Lagrange Transformations for Discrete Problems

In a similar way, discrete problems can first be transformed using a discrete version of the Lagrangian method. In this subsection, we present the Lagrangian transformations for discrete CSPs. The corresponding formulation for discrete optimization problems is similar and will not be shown.

Define $N(h(x))$, the norm in (3), as the number of unsatisfied constraints (conflicts) in $h(x)$, where $x \in Z$ for general integer problems and $x \in \{0,1\}$ for binary ones. The resultant problem formulation is as follows.

$$\begin{aligned} &\text{minimize} \quad N(h(x)) \\ &\text{subject to} \quad h(x) = 0, \quad x \in Z \end{aligned} \quad (20)$$

CSPs over discrete variables can be solved just like continuous problems as far as constraint satisfaction is concerned. The Lagrangian equation corresponding to (20) is

$$L(x,\lambda) = N(h(x)) + \lambda^T h(x) \quad (21)$$

The difference equations similar to (18) and (19) are

$$x_{k+1} = x_k - \Delta_x L(x_k, \lambda_k) \quad (22)$$

$$\lambda_{k+1} = \lambda_k + h(x_k) \quad (23)$$

where k is the iteration index, and Δ is the counterpart of gradient in discrete space. Here, $\Delta_x f(x)$ is a unit vector that reduces the value of $f(x)$ by the largest amount as compared to unit vectors in other directions.

It is easy to see that the necessary condition for (22) and (23) to converge is when $h(x) = 0$, implying that all the constraints are satisfied. If any of the constraints in $h(x)$ is not satisfied, the Lagrange multipliers will continue to evolve to handle the unsatisfied constraints.

As stated in Section 2.3, SAT problems defined in (7) can be considered as special cases of discrete CSPs. The difference equations corresponding to (22) and (23) for SAT problems are as follows.

$$x_{k+1} = x_k \ominus \Delta_x L(x_k, \lambda_k) \quad (24)$$

$$\lambda_{k+1} = \lambda_k + U(x_k) \quad (25)$$

Table 2. Global and local search components used in existing global optimization methods.

Method	Global Component	Local Component
Random Search	Uniform Sampling	Any Available Local Method
Genetic Algorithm	Selective Recombination	Optional
Simulated Annealing	Boltzmann Motion	Optional
Clustering Method	Cluster Analysis	Any Available Local Method
Bayesian Modeling	Bayesian Decision	Optional
Interval Method	Interval Calculation	Rarely Used
Covering Method	Informed Search with Bound Approximation	Rarely Used
Generalized Gradient	Traveling Trajectory	Rarely Used

where $\ominus$ in (24) denotes the exclusive-OR operator that sets x_{k+1} by flipping a chosen variable x_k corresponding to $\Delta_x L(x_k, \lambda_k)$. In this way, one of the x variables advances to its neighbor with Hamming distance 1. The Lagrange multipliers in (25) are updated according to the satisfaction of $U(x)$ in an individual basis.

3. Previous Work

In this section, we summarize previous approaches for solving constrained global optimization problems and discrete SAT problems. These are the two classes of problems addressed in later sections. At the level of global search, strategies for solving constrained problems are similar to those for solving unconstrained problems except in the handling of constraints.

3.1. Existing Global Optimization Strategies for Continuous Problems

A variety of deterministic and stochastic methods for finding global solutions to non-convex nonlinear optimization problems have been developed in the past three decades. A taxonomy on global optimization methods can be found in [94, 45, 24, 40, 63]. Figure 1 shows a classification of constrained global optimization algorithms surveyed in this section.

The survey that follows focuses on features of methods for solving continuous, constrained problems. Whenever it is possible, we analyze the balance that each algorithm strikes between global search and local refinement, and relate this balance to its performance. Table 2 summarizes this balance for a number of popular global optimization methods.

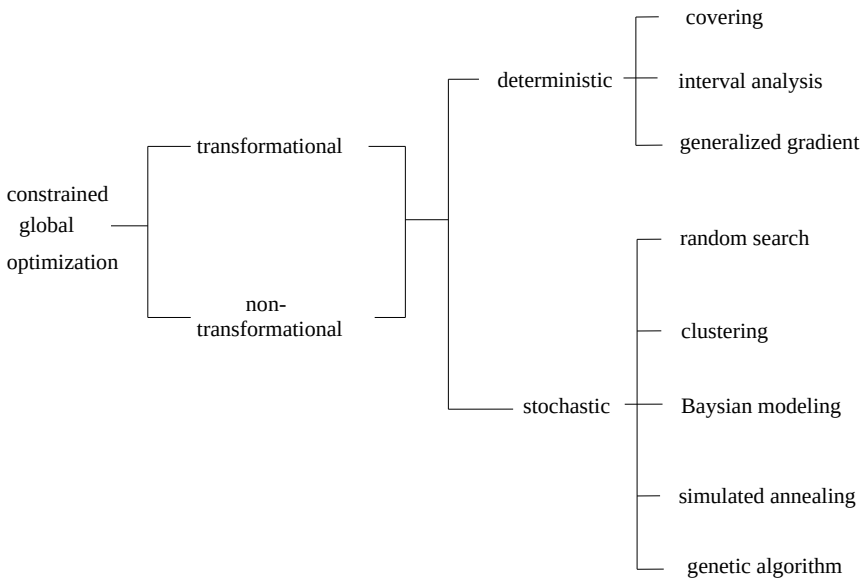

Figure 1. Classification of Global Optimization Algorithms

3.1.1. Stochastic Algorithms

A general stochastic algorithm [58, 94, 97, 72] for global optimization consists of three major steps [72]: a sampling step, an optimization step, and a step for checking the stopping criterion.

Random Search Methods. Three of the simplest random search algorithms are *pure random search*, *single-start* and *multi-start* [64, 100, 101, 41, 86]. They use random sampling for global search and a local search method to obtain the exact local optima. Their advantage is their simplicity and low sampling overhead. Since they are easy to implement, they are often used by non-experts to solve practical problems when global optimality is not critical. Their disadvantage, as pointed out in [94, 72, 69], is the excessive iterative steps spent in verifying whether the sampled points will eventually lead to new local minima. Hence, the design of stopping rules is critical in practical implementations [10, 8, 11, 9].

Genetic Algorithms (GA). Global search in GA involves the rational generation of new sample points based on performance of the whole population of samples in each iteration [25]. GA arrives at local solutions either as a result of gene drift at the end of the reproduction process or through local steps that are performed off-line. Gradient-like information is not used in the recombination process, and the effectiveness of applying GA to continuous closed-form problems remains to be justified [55]. This in turn limits GA in handling differentiable functions. Efforts were

reported in combining GA for global search and penalty functions for relaxing constraints to solve real-world engineering problems with closed form functions [66]. Using ten constrained optimization problems arising from mechanical, chemical, and electrical engineering areas, the results show that GA converges slowly and has difficulty in satisfying constraints.

Simulated Annealing (SA) is a simulation of the physical process of annealing, namely, the process of driving a physical system to a minimal energy configuration by means of a slow reduction in the temperature of the system. Its global strategy is derived from the concept of thermal equilibrium in a stochastic sense. Its key feature is in its controlled strategy for adaptive search that samples many regions of attraction before ending up as a greedy search. SA was originally designed to solve combinatorial optimization problems [50, 14]; for an extensive survey, see [1, 2]. The application of SA and related techniques to solve continuous global optimization problems can be found in [95, 13, 16, 65, 53, 47]. Recently, Romeijn and Smith [67] used SA to solve constrained continuous global optimization problems. Their results are comparable in quality to existing solutions on a collection of classical test problems.

Clustering Methods. By cluster analysis on random sample points, these algorithms try to start just one local search in each cluster in order to identify its local minimum. Two global strategies have been used for clustering [94]. The first [52] retains only points with relatively low function values to form clusters that correspond respectively to regions of attraction. The second [91] pushes each point towards a local minimum by performing a few steps of local descent. Classical treatments of clustering methods include [94, 52, 91], and more recent papers can be found in [92, 12, 90, 68, 94, 93]. Historically, clustering algorithms have been proposed to improve multi-start methods. Although they have some success in solving global optimization problems [90], they depend heavily on when to stop random sampling and switch to a local search. Late stopping will generate new sample points, leading to a very large number of extra local searches [72, 93].

Bayesian Methods schedule a new iteration of search in order to reduce the estimated risk of losing the global minimum, based on past observations of function values. Schagen [71] used a stationary stochastic process model to represent internally an objective function of success in a reasonable number of function evaluations. Based on Kushner's method [51] in one dimension, a global search algorithm for optimization in n dimension is presented in [88]. More introductions can be found in [94, 58, 97, 59]. A major drawback of Bayesian methods is their computational complexity that grows exponentially with the number of problem dimensions [88, 59]. Therefore, their use for solving multi-dimensional problems and for approximating objective functions is restricted [94, 97, 59]. They are believed to be most useful when the number of problem dimensions is up to around 15 [88].

3.1.2. Deterministic Algorithms

Instead of using sampling for global exploration, deterministic global optimization methods exploit the following global search strategies: (a) retain regions that contain good solutions while dropping regions that do not contain any solution (as in interval methods); (b) approximate the global solution iteratively by using tighter bounds (as in covering methods); (c) avoid getting trapped by reshaping function landscape or by imitating dynamic systems of particles in energy fields (as in generalized gradient methods).

Interval Methods [40, 60, 39] use interval analysis to eliminate regions containing no solutions. Eventually, a collection of solution intervals are found, some of which can have high quality solutions. To speed up interval analysis for differentiable problems, Newton-like and gradient-based methods can be used. Unfortunately, their complexity is extremely high, and the method is best suited for small problems.

Covering Methods. The simplest covering methods detect subregions not containing global minima and their exclusion from further consideration. For Lipschitz problems, covering methods provide the confidence in quality of solution in addition to the solution itself. Due to exponential complexity of covering algorithms, only problems of limited size (in the order of 10 to 20) can be solved. Acceleration techniques to improve performance has been proposed [6, 7]. A comprehensive treatment of covering methods is given in the monograph of Horst and Tuy [45]. A recent survey on covering methods can be found in [21].

Generalized Gradient Methods utilize gradient or higher-order information in such a way that continue the search trajectory every time a local solution is found. They date back to the 1960's and have many metaphors in physics. Classical works can be found in the survey book [94]. In general, this approach can be applied in two ways. First, the differential equation describing the search trajectory can be modified so that it can escape from local minima. These methods are a subclass of *trajectory methods* [96, 19, 70, 4, 81, 89, 3, 94]. Second, a standard local algorithm can be repeatedly applied to a modified function (such as the tunneling function [15] or filled function [26]). The modified function reshapes the original objective function by filling in dents that have been identified. As a result, the same local minima will not be revisited. These methods are a subclass of *penalty methods* [26, 15, 94].

3.1.3. Complexities of Global Optimization Methods over Continuous Variables

Global optimization of continuous nonlinear problems has been recognized as very difficult and intractable [62, 46]. The previous work surveyed in this subsection are generally heuristic in nature and depends heavily on problem formulations, initial starting points, and amount of time allowed.

Stochastic optimization algorithms are popular recently. They sample the objective function and perhaps compute the derivatives for a small number of points.

Since gradient information is not always available, the algorithm will not be able to know whether a function will dip to some unexpected small value between sample points [40]. Further, the required number of samples to arrive at a desired solution is often prohibitive for large problems.

On the other hand, deterministic algorithms (such as covering and interval analysis [40]) tries to guarantee the accuracy of solutions. As a result, they are forced to deal with severely restricted functions in order to exploit mathematically rigorous properties, and are not very useful for solving general nonlinear programming problems [94, 72, 45]. Compared with stochastic methods, very few deterministic methods have been used in practice [19].

In existing trajectory methods, intensive computations in global search prohibits their application to large problems. Generally, they do not employ good local search methods [54], such as conjugate gradient and Quasi-Newton methods. Rather, they use tightly coupled global and local search strategies, often compromising the global part by the local part. Note that the local part should focus on rates of convergence and/or constraint satisfaction, whereas the global part should emphasize on discovery of better or newer regions, avoiding revisits of the same local minima. This compromise between global and local searches severely limits the performance of global search, and results in very high computational complexity.

In view of the drawbacks in trajectory methods and the imbalance between global and local searches (Table 2), we propose in Section 4 a new algorithm that uses decoupled global and local search strategies. Our global search strategy is based on a traveling trace that collects geometrical information and uncovers new regions of local minima. Note that the functions we study are continuous and differentiable with possibly a finite number of discontinuities, and that each function has a set of regions of attraction, each of which is a continuum with gradually changing contours. Consequently, if the trace passes through the vicinity of a local minimum, a local search will be able to identify it. By separating global from local searches, our method is able to identify local minima more effectively than previous trajectory methods.

3.2. Related Works on SAT Problems

Previous methods for solving SAT problems can be classified into discrete and continuous, the latter involving the transformation of a discrete SAT problem into a continuous problem before solving it.

3.2.1. Discrete methods

Discrete methods can be either complete or incomplete, depending on their ability to prove infeasibility. Complete methods use some form of informed backtracking to search the space systematically, whereas incomplete methods usually rely on ad hoc heuristics. Complete search methods for solving SAT problems include resolution

and Davis-Putnam [18] procedure. They are computationally intensive because they are enumerative in nature. For instance, Selman *et al.* [75] and Gu [35] have reported that Davis-Putnam procedure cannot handle SAT problems with more than 150 variables.

Incomplete discrete search methods are generally based on random restarts and methods to evaluate a current (partial) assignment. In random restarts, a local search is first performed at a starting point, and a new starting point is generated when no further improvement is found from the previous starting point. The evaluation of whether a state is promising is done by the evaluation function. Examples of which include probabilistic measures (as in simulated annealing [50, 14]), heuristic functions (as in fitness functions of genetic algorithms [43, 55]), and number of constraints violated (as in constraint-satisfaction algorithms). The problem with random restarts is that a search in a seemingly good direction may get stuck in a very small local minimum, and a random restart may bring the search to a completely different search space. Selman *et al.* [76] has found that annealing is not effective for solving SAT problems. To the best of our knowledge, there is no successful application of genetic algorithms to solve SAT problems.

Recently, some local search methods were proposed and applied to solve large SAT problems [61, 27, 17]. The most notable ones are those developed independently by Gu and Selman.

Gu developed a group of local search methods for solving SAT and CSP problems. In his Ph.D thesis [29], he first formulated conflicts in the objective function and proposed a discrete relaxation algorithm (a class of deterministic local search) to minimize the number of conflicts in these problems. The algorithms he developed subsequently focused on two components: methods to continue a search when it reaches a local minimum, and methods for variable selection and value assignment. In the first component, he first developed the so-called *min-conflicts* heuristic [29] and showed significant performance improvement in solving large size SAT, n-queen, and graph coloring problems [29, 83, 84, 82, 85]. His methods use various local handlers to escape from local traps when a greedy search stops progressing [30, 36, 37, 31, 38, 32]. Here, a search can continue without improvement when it reaches a local minimum [36] and can escape from it by a combination of backtracking, restarts, and random swaps. In variable selection and value assignment, Gu and his colleagues have developed random and partial random heuristics [30, 83, 36, 84, 82, 37, 33, 31, 38, 34, 35]. These simple and effective heuristics significantly improve the performance of local search algorithms by many orders of magnitude.

Selman developed GSAT [79, 75, 76, 78, 73, 77] that starts from a randomly generated assignment and performs local search iteratively by flipping variables. Such flipping is repeated until either a satisfiable assignment is found or a pre-set maximum number of flips is reached. When trapped in a local minimum, GSAT either moves up-hill or jumps to another random point. To avoid getting stuck on a *plateau*, which is not a local minimum, GSAT makes side-way moves.

In short, the objective function in (6) may have many local minima that trap local search methods. Consequently, a search in a seemingly good direction may get

stuck in a small local minimum, and will rely on random restarts or hill climbing to bring the search out of the local minimum. However, both schemes do not explore the search space systematically, and random restarts may bring the search to a completely different search space.

SAT can also be considered as a *constraint-satisfaction problem* (CSP). Existing approaches to solve constraint-satisfaction problems include backtracking [56, 34], best-first search, most-constrained first search [87], and local-search methods such as hill-climbing [57, 76, 75, 56, 34]. These methods are generally combined with heuristic guidance such as conflict minimization [56, 85].

3.2.2. Continuous methods

In the continuous approach, discrete variables in the original SAT problem are first transformed into continuous variables. After finding a solution in the continuous space, the continuous variables are restored into integers from fractional values. Direct transformation and functional transformation are two ways to transform discrete variables into continuous ones. The latter uses a continuous function that resembles a step function (such as a sigmoid function), whereas the former formulates the problem in such a way that a feasible solution in the transformed problem is the same as that in the original problem.

After transforming a discrete problem into a continuous one, various local-search methods can be applied to solve it. These include simplex methods for solving linear programming problems, steepest descent, conjugate gradient, Quasi-Newton, and Lagrange-multiplier methods [54]. For instance, Hopfield-type neural networks [44] are a steepest descent implementation based on a set of differential equations. Another approach proposed by Gu [33, 34, 35] is to use direct transformation and apply descent methods such as steepest descent and conjugate gradient.

The disadvantage of searching in the continuous space is that continuous solutions found are not guaranteed to satisfy the original constraints after restoring the continuous variables to integral ones. Moreover, searches in the continuous space often get trapped in local minima, requiring multi-starts or backtracking to restart the search from a new starting point. One exception in which a combined local search and backtracking is effective was recently proposed by Gu [35], who transformed a SAT problem into an unconstrained global optimization problem in real space. Using a combination of gradient descents in continuous space and backtracking in discrete space, Gu showed significant improvements in solution time for solving certain classes of conjunctive normal-form formulae. For other problems, a local search often leads to local minima, and the number of times that the algorithm backtracks grows exponentially with problem size.

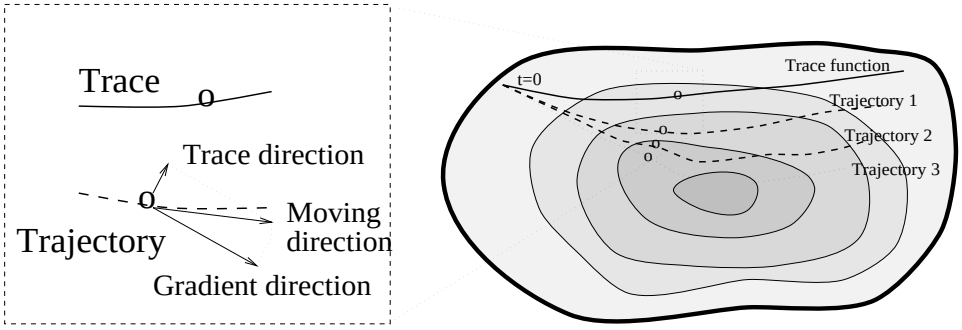

Figure 2. Global search and local refinement are the two phases of *NOVEL*, whose behavior is demonstrated by the trajectory plot on the Lyupunov contour map. In the global search phase, the trajectory shows a combined effect of gradient descents and pull exerted by the moving trace. In the local search phase, the trajectory is sampled to collect starting points for pure local descents.

4. Solving Constrained Global Optimization Problems by NOVEL

In this section, we present *NOVEL* for solving various nonlinear optimization problems modeled in (8) and CSPs modeled in (20). In both cases, we first relax the constraints using Lagrange multipliers into (12) and (21) before applying global search. We illustrate the interface between our global search method and existing local optimization methods. Finally, we discuss a special type of trace that is suitable for constraint satisfaction problems.

4.1. NOVEL for Constrained, Continuous Problems

NOVEL is a global optimization method with three major features: exploring the problem space, identifying promising regions, and pinpointing exact locations of local optima.

In exploring the search space, the trace plays an important role in uncovering regions with new local minima. A *trace* is a continuous aperiodic function of (logical) time that generates a *trajectory*. At time 0, both the trace and the trajectory start at the same point. As the trace moves from point x_1 to point x_2, the trajectory moves from point y_1 to y_2, where y_2 is a function of the local gradient at y_1 and the distance between x_1 and y_1 (see Figure 2). These two counteracting forces, descents into local minima and attraction exerted by the trace, form a composite vector that represents the route taken by the trajectory.

Note that a trajectory generated in one run can serve as the trace function in the next run, thereby allowing the trajectory to eventually converge to local minima. However, this multi-stage application is unduly inefficient as it may take a large

number of stages before convergence is reached. As a result, promising points on the trajectory can be identified, and local descents using existing methods can be applied to find good local minima. This concept is also illustrated in Figure 2.

NOVEL is a systematic method to find new, unvisited promising regions of attraction without losing good regions found earlier. Its equation for constrained minimization is extended from (15) and (16) and can be described as follows.

$$\frac{dx}{dt} = -\bigtriangledown_x \mathcal{L}(x(t),\lambda(t)) + w * (\mathcal{L}_x(x(t),\lambda(t)) - g_x(x(t),\lambda(t))) \quad (26)$$

$$\frac{d\lambda}{dt} = \bigtriangledown_\lambda \mathcal{L}(x(t),\lambda(t)) + w * (\mathcal{L}_\lambda(x(t),\lambda(t)) - g_\lambda(x(t),\lambda(t))) \quad (27)$$

where $g_x(t) = (g_{x_1}(t), g_{x_2}(t), \ldots, g_{x_n}(t))$ and $g_\lambda(t) = (g_{\lambda_1}(t), g_{\lambda_2}(t), \ldots, g_{\lambda_m}(t))$ are trace functions for the original and Lagrange variables, respectively, w is a weight, and $\mathcal{L}$ is defined in (12). Note that in (26) and (27), trace functions $g_x(x(t),\lambda(t))$ and $g_\lambda(x(t),\lambda(t))$ have been added to the right-hand sides of (15) and (16).

It should now be clear that, instead of using restarts, *NOVEL* uses a continuous trace to travel through a problem space in order to produce a terrain-specific trajectory of $(x(t),\lambda(t))$. A trace can be considered as a terrain-independent trajectory that guides the global search in the solution space. A good trace should be aperiodic so that it does not return to the same starting point and regenerates possibly the same trajectory. It needs to be continuous in order to be differentiable. This allows the trajectory generated to follow the terrain in a continuous manner without restarting to new starting points. It should be bounded so that it will not explore unwanted regions. Finally, it should be designed to travel from coarse to fine so that it examines the search space in greater details when more time is allowed.

Since the design of a trace function is an intractable functional programming problem, we have studied a number of heuristic functions and fine-tuned them experimentally. One of the best functions we have found is as follows.

$$g_i(t) = \rho\, sin\left[2\pi\left(\frac{t}{2}\right)^{0.95-\frac{0.45(i-1)}{n})} + \frac{2\pi(i-1)}{n}\right] \quad (28)$$

where $i = 1, \ldots, n+m$ and ρ defines the search range.

Eq. (28) is a mechanism to bring the search trajectory out of local minima and leads the search to other regions. Without it, a gradient-based method will lead the trajectory to a local minimum, and will not be able to bring it out unless the search is restarted. This global-local balance in *NOVEL* is controlled by weight w in (26) and (27). In general, w is a time-varying function that adapts to the changing terrain seen by the trace. More weight on the trace makes the corresponding trajectory follow the trace closely, thereby reducing local descents. This is useful when the local slope is large. On the other hand, less weight is desirable when the slope is small, allowing the trajectory to better explore the local terrain. In this paper, we have chosen a constant weight for simplicity. Our experiments show

that *NOVEL* is very robust with respect to a large range of w for the application problems we have studied.

Note that (26) and (27) do not pinpoint the exact locations of local minima. Rather, a set of starting points need to be identified from the trajectory, and any existing local search algorithm can be applied from these starting points to find local minima.

NOVEL is more efficient than multi-start algorithms because it makes informed decisions based on the information-bearing trajectory before applying expensive descents to find local solutions. This greatly reduces the chance of redetermining local solutions already found. *NOVEL* is more efficient than sampling in stochastic methods because its trajectory is a continuous probe to the problem space and will less likely miss a region of attraction in between surrounding regions already found. Finally, experimental results show that the time spent on generating the trajectory and sampling it is very small as compared to the time to do one local descent. This is a small price to pay for improved quality.

An important point worth noting is that in the space spanned by the original and the Lagrange variables, (18) and (19) will be attracted to a saddle point inside the divides that encompass the initial point. Hence, finding a satisfying assignment by *NOVEL* is a local search in this space, consisting of descent in the original variable space and ascent in the Lagrange variable space. Changing the initial point only affects where local optimization is started. As long as there is at least one satisfying solution within a promising region surrounded by its divides, any starting point selected will lead the trajectory to the solution in this region. Note that a local search without restarts in the original-variable space alone will usually get stuck in local minima and is *not sufficient* to find solutions that satisfy the constraints.

Another distinct feature of *NOVEL* is that it uses repair heuristics that starts from a complete but unsatisfying assignment, and tries to repair it in order to reduce the number of conflicts using optimization techniques based on continuous or pseudo-continuous trajectories. Minton *et al.* [56] have noted that repair-based heuristics guide an optimization process with a larger picture about the current solution state that is not available to standard backtracking algorithms.

For Lagrangian formulations of constrained problems, we need a merit function to measure how close a trajectory is from a saddle point (a local minimum that satisfies the constraints). Here we use a Lyupunov function to identify points to use in the trajectory for local descents. Lyupunov functions have been widely used in applications such as automatic control, neural networks and nonlinear optimization to measure solution quality. The Lyupunov function $F(x, \lambda)$ is defined as

$$F(x,\lambda) = || - \bigtriangledown_x \mathcal{L}(x,\lambda)||^2 + ||\bigtriangledown_\lambda \mathcal{L}(x,\lambda)||^2. \tag{29}$$

It takes a value of 0 at a saddle point and is positive elsewhere. Hence, it can be used as an indicator of the distance from (x, λ) to a saddle point. Note that $F(x(t), \lambda(t))$ is a one-dimensional, multimodal function of time t.

We can now put all the parts of *NOVEL* together to solve the constrained continuous optimization problem defined in (8).

1. *Transformation.* Apply the augmented Lagrangian transformation defined in (12).

2. *Global Search Phase.* Solve the system of ordinary differential equations (26) and (27). There can be multiple stages of this step in which the trajectory of a stage is used as the trace function in the next stage.

3. *Sampling Phase.* Compute the Lyupunov values using (29) as a function of (logical) time. Pick a set of starting points from local minima of the Lyupunov values.

4. *Local Search Phase.* From a starting point, solve the original problem (8) using the augmented Lagrangian formulation (12). This involves solving the system of differential equations (13) and (14). This step stops when a saddle point has been reached.

We illustrate the algorithm by applying it to solve a problem whose objective function is Levy's No. 3 function and whose constraint is an elliptic function.

$$\begin{aligned} \text{minimize} \quad & (\cos(1) + 2\cos(2+x) + 3\cos(3+2x) + 4\cos(4+3x) \\ & + 5\cos(5+4x)) * (\cos(1+2y) + 2\cos(2+3y) \\ & + 3\cos(3+4y) + 4\cos(4+5y) + 5\cos(5+6y)) \\ \text{subject to} \quad & -0.75 + 0.883883\,x + 3.90625\,x^2 - 0.883883\,y \\ & + 4.6875\,x\,y + 3.90625\,y^2 \leq 0 \end{aligned} \tag{30}$$

Figures 3a-3b depict the terrain in the vicinity of the constrained region. They show that the terrain is not rugged and the solution is relatively easy to find. Figure 4a shows a minimum in the feasible region that is bordered by two regions of attraction of two other minima. The plots of the Lagrangian function (Figures 4c and 4d) show that many local minima of the objective function have been eliminated, and that global search in the space of the original and Lagrange variables is more effective than that in the original-variable space alone. Finally, the plots show that the trace and the trajectory start at the same point, that the trace does not enter the feasible region, and that the trajectory eventually stops in the feasible region. We did not show the local search phase due to the simplicity of the example.

4.2. Adaptation of NOVEL to solve CSPs

In this section, we show how CSPs formulated using the Lagrangian method can be solved by *NOVEL*.

For continuous CSPs, the Lagrangian formulation with an artificial merit function (17) and the corresponding differential equations (18) and (19) can be rearranged

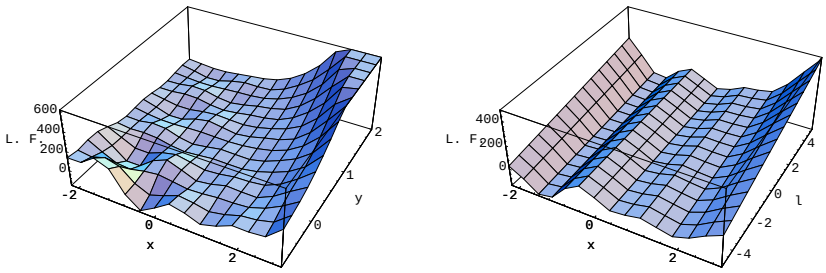

Figure 3. A continuous constrained optimization problem with Levy's No. 3 function as its objective and an elliptic function as its constraint. This figure shows 3-D plots of Lagrangian function $\mathcal{L}$ on dimensions (a) x and y and (b) x and λ.

into the following form.

$$\frac{dx}{dt} = -\nabla_x N(h(x)) + g(t) \tag{31}$$

$$\text{where } g(t) = -\left[\int_0^t h(x(\tau))\, d\tau\right] \nabla_x h(x(t)) \tag{32}$$

$g(t)$, a time varying function, is based on (19) that relates the Lagrange multipliers and $\nabla_x h(x)$, the Jacobian of $h(x)$. We call $g(t)$ an *implicit trace* because it is a trace determined by the system at run-time rather than defined ahead of time.

There are two forces in the dynamics of (31). On one hand, the trajectory tries to follow a gradient path of $N(h(x))$ in order to find a solution satisfying $N(h(x)) = 0$. On the other hand, $g(t)$ in (32) will bring the trajectory $x(t)$ out of any local minimum when $N(h(x))$ is larger than 0. As shown in (19), $g(t)$ will change over time as long as $h(x)$ is not zero. Consequently, $g(t)$ will pull the trajectory out of local traps and lead the trajectory to a neighboring region without restarting randomly. The direction in which the trajectory (32) will go depends on the history of constraint violation $\int_0^t h(x(\tau))\, d\tau$ and the information returned by $\nabla_x h(x)$.

Our experiments show that (31) and (32) are effective to lead the trajectory out of local minima. Moreover, it carries all the benefits of the original method in using a continuous trajectory to traverse from one region to another without relying on random restarts. Note that in solving CSPs using (31), local optimization is not required, as the first satisfiable solution is a solution in the Lagrangian space of the original CSP modeled by (18) and (19).

So far, we have discussed the application of *NOVEL* for continuous CSPs. For discrete SATs, we start with a discrete Lagrangian formulation (21) and generate a pseudo-continuous trajectory that moves from one discrete point to another. By rearranging (22) and (23), we get a set of finite difference equations.

$$x_{k+1} = x_k - \Delta\, N(h(x_k)) + g_k \tag{33}$$

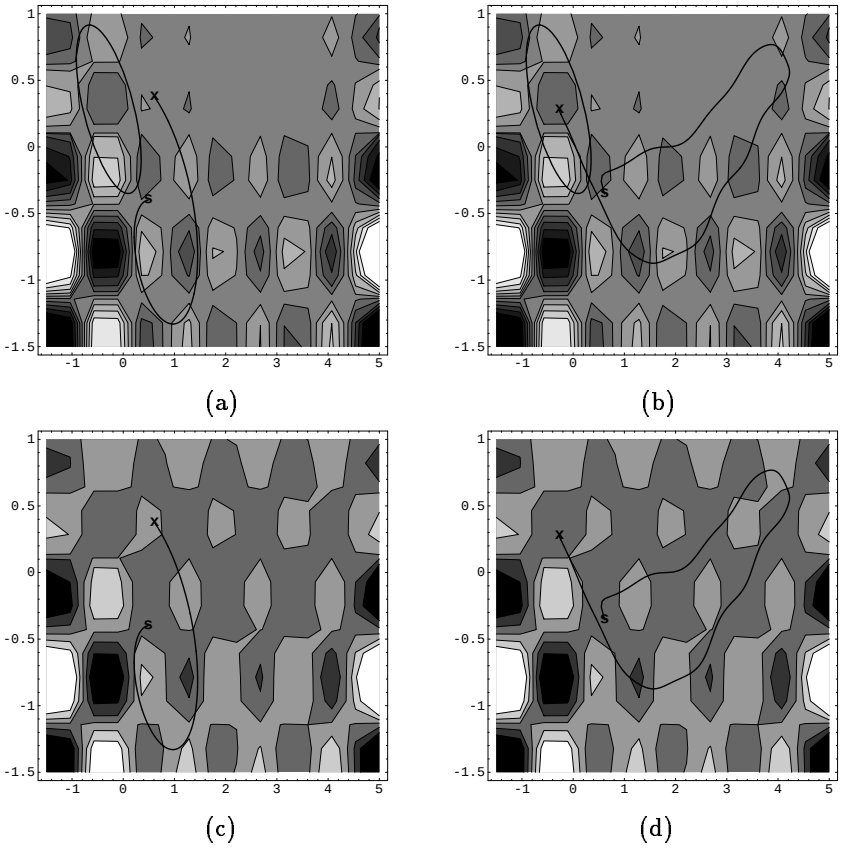

Figure 4. 2-D contour plots of the objective function for the example in Figure 3 showing (a) the trace and (b) the corresponding trajectory; 2-D contour plots of the Lagrangian function showing (c) the trace and (d) the corresponding trajectory. Here, S and X are the starting and ending points, respectively.

$$\text{where } g_k = -\left[\sum_{l=0}^{l-1} h(x_l)\right] \Delta h(x_k) \tag{34}$$

Here, g_k is the implicit trace function that accounts for both its history of constraint violation, $\sum_{j=0}^{l-1} h(x_l)$, and its current status of constraint satisfaction, $\Delta h(x_k)$. Note that $\Delta h(x)$ is an m-by-n matrix where m is the number of variables, n is the number of constraints, and its $(i,j)^{\text{th}}$ entry is 0 if $h_i(x) = 0$ and $\frac{\partial h_i(x)}{\partial x_j}$ otherwise.

Finally, we summarize in Figure 5 the mathematical formulations of methods derived in Sections 2.5 and 4.1 for solving unconstrained continuous problems versus constrained continuous problems. These equations are further classified into those using local optimization techniques versus those using global optimization techniques. The figure clearly indicates that *NOVEL* is a concept that can be

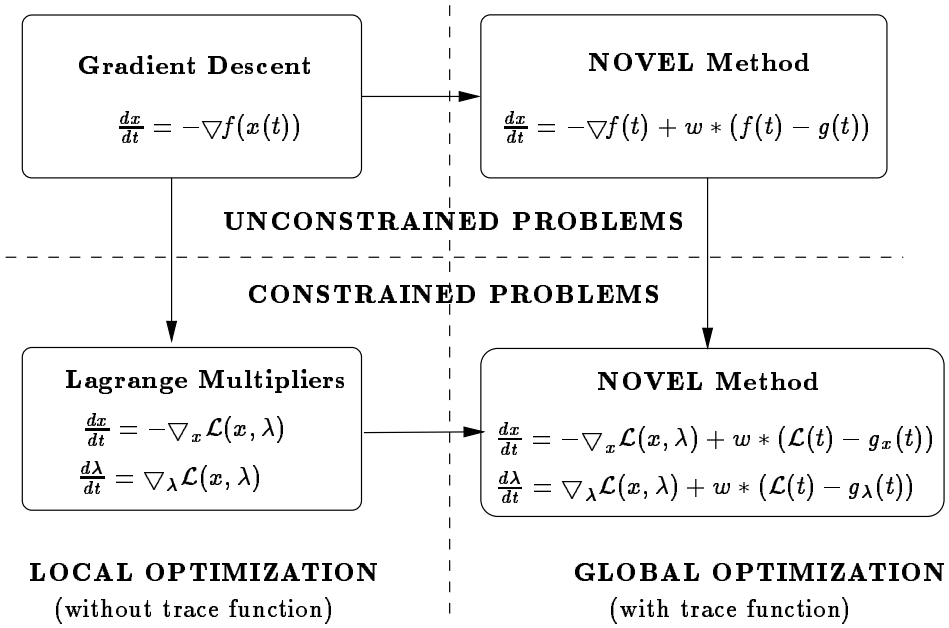

Figure 5. Relationship between *NOVEL* and local optimization methods.

integrated easily into existing local optimization methods. Equations for discrete problems are similar and will not be shown.

5. Results on Continuous Constrained Optimization Problems

In this section we describe experimental results on some existing constrained optimization benchmarks [23] (Table 1). These benchmarks are challenging because they model practical applications that have been studied extensively in the past. As a result, improvements are generally difficult. The problems we have studied in this paper are of moderate sizes; results on larger problems will be reported in the future. Results on continuous unconstrained problems are reported in [80].

5.1. Implementation: The NOVEL Package

We have developed a package to implement *NOVEL*. The package has a Kernel, a Parser/Translator, and a Visualization Frontend. Figure 6 shows its structure.

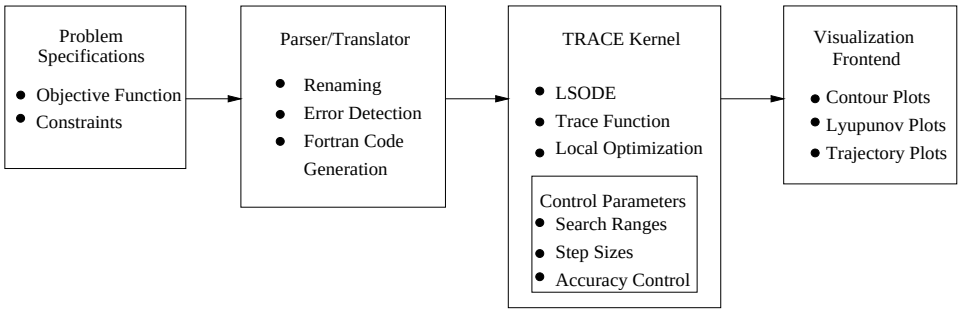

Figure 6. Organization of the *NOVEL* package.

NOVEL is a global optimization system for solving general constrained and unconstrained problems. Users specify an objective function and a constraint set in symbolic form as well as some control parameters.

The specified problem is supplied to the Parser/Translator. The Parser checks the input for integrity, detecting typographical errors, runaway variables, and some syntax errors. Any error encountered will be reported. The parser then renames all the variables entered and assigns them to array entries. (This is essential for efficiency reasons.) Automatic renaming allows variables to be entered as they were specified. The Parser also accepts high-level directives to simplify data entry. For instance, to declare all variables to be positive, the directive "$VariablePositive \leftarrow True$" can be used instead of declaring the lower bound of each variable to be zero. Last, the Parser provides problem statistics, such as the number of variables, constraints, and bounds when parsing is finished.

If parsing is successful, a Symbolic Translator will translate the objective function and constraints and produce the Lagrangian function and its gradients needed in the optimization process. The translator is currently written in *Mathematica*[1]. Since differentiation is done symbolically, the gradients are in closed forms and can be computed accurately in any computer. Finally, the Translator generates the code into Fortran to be compiled into the Kernel.

The major part of the Kernel is LSODE[2], a package for solving ordinary differential equations. It works on a variety of platforms, as all the programs are coded in Fortran.

Finally, results output by the Kernel are displayed in graphical form using the Visualization Frontend. For problems that need rescaling or have numerical instability, an off-line sensitivity analyzer will be used to verify constraint feasibility and detect ill-scaled or hard constraints.

5.2. An Illustrative Example

We illustrate *NOVEL* by solving a constrained optimization problem and by showing the solution process graphically in the global search phase.

The following engineering problem (Problem 5.2.1 in [23]) is a distillation sequencing problem of non-sharp separation with a three-component feed mixture that has to be separated into two three-component products. This is a nonconvex nonlinear programming problem with 48 variables, 13 linear constraints, and 25 nonlinear constraints. All the variables take only nonnegative values.

$$
\begin{aligned}
\text{minimize} \quad & f = a01 + (a11 + a21\, rlkA1 + a31\, rhkB1 + bA1\, xA5 + bB1\, xB5)\, F5 + \\
& a02 + (a12 + a22\, rlkB2 + a32\, rhkC2 + bA2\, xA13 + bB2\, xB13)\, F13 \\
\text{subject to} \quad & F1 + F2 + F3 + F4 - 300 = 0,\ F6 - F7 - F8 = 0 \\
& F9 - F10 - F11 - F12 = 0,\ F14 - F15 - F16 - F17 = 0 \\
& F18 - F19 - F20 = 0,\ F6\, xA6 - rlkA1\, fA5 = 0 \\
& F14\, xB14 - rlkB2\, fB13 = 0,\ F9\, xB9 - rhkB1\, fB5 = 0 \\
& F18\, xC18 - rhkC2\, fC13 = 0,\ fA5 - F5\, xA5 = 0 \\
& fB5 - F5\, xB5 = 0,\ fC5 - F5\, xC5 = 0 \\
& fA13 - F13\, xA13 = 0,\ fB13 - F13\, xB13 = 0 \\
& fC13 - F13\, xC13 = 0,\ fA5 - F6\, xA6 - F9\, xA9 = 0 \\
& fB5 - F6\, xB6 - F9\, xB9 = 0,\ fC5 - F6\, xC6 - F9\, xC9 = 0 \\
& fA13 - F14\, xA14 - F18\, xA18 = 0,\ fB13 - F14\, xB14 - F18\, xB18 = 0 \\
& fC13 - F14\, xC14 - F18\, xC18 = 0,\ 0.333F1 + F15\, xA14 - fA5 = 0 \\
& 0.333F1 + F15\, xB14 - fB5 = 0,\ 0.333F1 + F15\, xC14 - fC5 = 0 \\
& 0.333F2 + F10\, xA9 - fA13 = 0,\ 0.333F2 + F10\, xB9 - fB13 = 0 \\
& 0.333F2 + F10\, xC9 - fC13 = 0 \\
& 0.333F3 + F7\, xA6 + F11\, xA9 + F16\, xA14 + F19\, xA18 - 30 = 0 \\
& 0.333F3 + F7\, xB6 + F11\, xB9 + F16\, xB14 + F19\, xB18 - 50 = 0 \\
& 0.333F3 + F7\, xC6 + F11\, xC9 + F16\, xC14 + F19\, xC18 - 30 = 0 \\
& xA5 + xB5 + xC5 - 1 = 0,\ xA6 + xB6 + xC6 - 1 = 0 \\
& xA9 + xB9 + xC9 - 1 = 0,\ xA13 + xB13 + xC13 - 1 = 0 \\
& xA14 + xB14 + xC14 - 1 = 0,\ xA18 + xB18 + xC18 - 1 = 0 \\
& rlkA1 \le 0.85,\ rlkB2 \le 0.85,\ rhkB1 \le 0.85,\ rhkC2 \le 0.85 \\
& rlkA1 \ge 1,\ rlkB2 \ge 1,\ rhkB1 \ge 1,\ rhkC2 \ge 1 \\
\text{Initial} \quad & a01 = 0.23947; a02 = 0.75835; a11 = -0.0139904; a12 = -0.0661588; \\
\text{Conditions:} \quad & a21 = 0.0093514; a22 = 0.0338147; a31 = 0.0077308; a32 = 0.0373349; \\
& bA1 = -0.0005719; bA2 = 0.0016371; bB1 = 0.0042656; bB2 = 0.0288996; \\
& xC6 = 0; xA18 = 0; xC6 = 0; xA18 = 0
\end{aligned}
$$

The test problem is solved with $w = 10$ and a search range from -10 to 10 using the initial assignment specified in [23]. We have used a three-stage cascaded implementation in which there are three applications of *NOVEL*, with the output of one stage feeding into the input of the next stage. In Figure 7, we plot the Lyupunov function values for the trace and the trajectories.

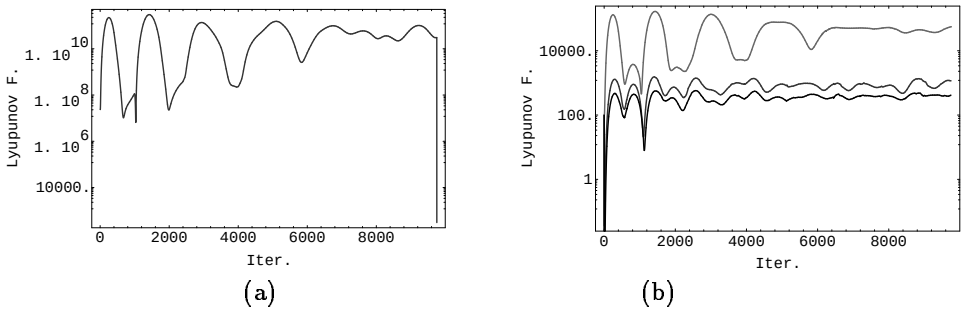

Figure 7. Lyupunov values of the trace function and the trajectories of a 3-stage cascaded process observed in solving the example. (a) Lyupunov values of the trace function with respect to logical time; (b) Lyupunov values of the trajectories of the three cascaded stages, with the last stage having lower Lyupunov values.

Since the trace is terrain-independent and may be far away from feasible solutions, its Lyupunov values are generally large (Figure 7a). By combining gradient information for the original and Lagrange variables (x,λ), the trajectory moves closer to the local minima. This is depicted in Figure 7b that shows the Lyupunov values for the three trajectories. It further shows that increasing the number of stages is beneficial in drawing the trajectory closer to the local minima. Note that there is a trade-off between solution quality and computation time when deciding on the number of stages to use. If the terrain is very rugged, then it is beneficial to use more stages, as the global search phase is more effective in overcoming local minima. On the other hand, if the terrain is smooth, then using more stages will not improve the solutions. Our experience shows that three cascaded stages are often enough for most applications we have studied. In a few cases, even using one stage will result in the same solution quality.

For a thorough study of the behavior of *NOVEL*, we allocated 4 days of CPU time to solve the test problem. We obtained in 90 minutes a solution of value -1.05 which is better than the best known value of 1.56 reported in [23]. Prior to obtaining this solution, 27 calls to the Lagrange local optimization routine were made. Further, all the 27 solutions found by these calls resulted in the same solution reported in [23]. No further improvement was found by *NOVEL* after getting the better solution of -1.05.

5.3. Experimental Results

We have applied *NOVEL* to solve a number of engineering problems in [23] and have compared the results to those reported there. For reasons stated in Section 4, we use the same parameters (step size and trace function) to solve all the problems. We have also used initial points that *NOVEL* started, as suggested in the benchmark set. The only exception is the search range which has to be modified for each

Table 3. Results of some engineering design problems using *NOVEL* with a comparison to the best known solutions [23]. *NOVEL*'s time is in seconds on a Sun SS 10/51 computer.

Prob. ID	Application	# Var.	# Const.	# Probes	Time (sec.)	*NOVEL*	Known
3.1.1	nonconvex	8	6	11173	49704	7049	7049
3.2.1	nonconvex	4	6	179	16428	-30665	-30665
4.5.1	test NLP	15	6	798	681	-13.45^*	-11.9 6
4.8.1	heat exchanger	7	8	1888	70314	189.3	189. 3
4.9.1	heat exchanger	11	9	6	52616	288^*	7049
5.2.1	distillation	46	36	655	277639	-1.05^*	
5.3.1	distillation	167	62	5	84000	2.06^*	2.98
5.4.1	pooling	38	32	219	263419	1.86	1.86
6.2.1	pooling	9	6	780	17936	400	400
6.3.1	pooling	11	6	603	17862	600	600
6.4.1	pooling	11	6	928	1862	750	750
7.2.1	heat exchanger	16	13	4407	541255	56511^*	56825
7.3.1	heat exchanger	27	19	3713	56730	45370^*	46254
7.4.1	heat exchanger	38	23	718	312579	10419.8^*	34633

specific problem. The reason for not tuning the parameters is to avoid any bias, as good solutions can always be obtained by sufficient tuning. We used three cascaded stages in *NOVEL* in generating these results.

Table 3 summarizes the results found. (Results for some of the smaller problems are not reported unless there are improvements; results for the larger problems are incomplete at this time.) Column 1 lists the problem identifications that appear in the benchmark collection [23]. Columns 2, 3, and 4 give the application domains, the number of variables, and the number of constraints of each test problem.

The column labeled '# probes' denotes the number of starting points sampled from the trajectory in the third stage in the time that *NOVEL* was run. These points were used as starting points from which local constrained optimization was performed. The number of starting points depends on the number of dents that appeared in the corresponding Lyupunov curve of the trajectory. Note that some of these starting points may lead to the same solution, and starting points that appear far apart may actually lie in the same region containing one local minimum. This situation can only be discovered when local optimization is run. We have observed this phenomenon in Problems 4.6.1, 5.2.1, 6.4.1, 7.2.1, and 7.3.1, and have found it a rule rather than an exception: the more time the global search was run, the more frequently such a revisit could happen. It is not unusual that the last 20% of improvement calls for 80% of the total time.

Column 6 shows the CPU time spent on each problem. The time limit is set in an ad hoc fashion because the relationship between solution quality and computation time is problem dependent. For problems 5.2.1 and 7.2.1, solutions with quality as good as 99% of those shown in Table 3 were obtained within 477 and 15 seconds, respectively. However, we did not stop the search as that point as we did not know whether better solutions would be obtained when given more time.

Procedure NOVEL
Generate random truth assignment x
while $u(x) > 0$
    Compute $\Delta_x L$
    $x = x \ominus \Delta_x L(x, \lambda)$
    Compute $U(x)$
    $\lambda = \lambda + U(x)$
end while

Function $\Delta_x L(x, \lambda)$
$min\_L = \infty$
for $i := 1$ **to** m
    Flip $x[i]$
    Compute $L(x, \lambda)$
    if $L(x, \lambda) < min\_L$
    then
        $min\_L = L(x, \lambda)$
        $min\_position = i$
    end if
    Restore $x[i]$
end for
for $i := 1$ **to** m
    $\Delta_x L(x[i], \lambda) = 0$
end for
$\Delta_x L(x[min\_position], \lambda) = 1$
return $\Delta_x L(x, \lambda)$

Figure 8. Pseudo-code of *NOVEL* for SAT.

Column 7 shows the results obtained by *NOVEL*, and Column 8, the best known results to date. Results with an asterisk have been improved over previous results, where improvements range between 1% to 2,400%. We did not find any improvements for Problems 3.1.1, 3.2.1, 4.8.1, 5.4.1, 6.2.1, 6.3.1, 6.4.1. This may be caused by the fact that the solutions obtained by existing methods are already very good, or simply more time is required to improve the solution. Our results indicate that *NOVEL* is very robust in discovering new regions and in escaping from local traps.

Results on applying NOVEL to optimize filter bank design are presented elsewhere [99].

6. Experimental Results on SAT Problems

In this section, we apply a discrete version of *NOVEL* to solve SAT problems. Using (20), we first transform a SAT problem into a constrained optimization problem with a heuristic objective of minimizing the number of unsatisfied clauses. The pseudo code based on (24) and (25) is shown as follows.

Given m variables and n clauses in a SAT problem, the complexity of the pseudo code is as follows. An update of λ, involving the computation of $U(x)$, takes $O(n)$ time, assuming that the number of literals in each clause is less than a constant k. This is less than the $O(mn)$ complexity of computing $\Delta_x L(x, \lambda)$; hence, the order-of-magnitude complexity remains unchanged. Computing $\Delta_x L(x, \lambda)$ differs

Table 4. Results on circuit diagnosis problems comparing *NOVEL*'s results on Sun SS 10/51, GSAT's results on SGI Challenge [76, 74], and Fleurent *et al.* tabu search on Sun SS 10/50 [22].

Problem			*NOVEL* (Max-Flips=625 per trial)					GS [76]	GS [74]	Tabu
ID	Var.	Clauses	Trials	Time (sec.)			Clauses	Time	Time	Time
			avg	max	min	avg	Unsat'd	(sec.)	(sec.)	(sec.)
ssa7552-158	1363	3034	428	25	7	15	0	90	35*	22
ssa7552-038	1501	3575	541	22	10	18	0	129	10	16
ssa7552-159	1363	3032	371	18	8	12	0	720	4	91
ssa7552-160	1391	3126	323	14	7	10	0	N/A	3	69

* Results excluding 2 failures in 10 trials.

from a greedy step only in the use of L instead of $u(x)$, and has complexity $O(mn)$. Empirically, the new algorithm takes around 30% more overhead as compared to its greedy version.

A typical scenario in the trace-based optimization goes as follows. In the original-variable space, the trajectory of $N(h(x))$ goes up and down and visits the region where the current assignment is in. If a satisfying assignment such that $N(h(x)) = 0$ is in this region, then the solution will be found by gradient descent, and the implicit trace due to the Lagrange multipliers (34) will settle down as less violation is seen along the way. If the region contains no satisfying assignment, then the implicit trace due to the Lagrange multipliers (34) will be changing according to the deviation of $h(x)$ from 0. In this case, the trajectory will get out of the region with the help of Lagrange variables that guard against constraint violation.

We have evaluated *NOVEL* using SAT benchmarks in the DIMACS archive. Due to the large number of instances, we have only evaluated the following benchmarks.

- **Circuit Diagnosis Problems**. Alan Van Gelder and Yumi Tsjuji contributed a set of SAT formulas based on circuit fault analysis.
- **Boolean Inductive Problems**. Kamath *et al.* [49] developed a set of SAT encodings of Boolean induction problems. The task is to synthesize (or induce) a logical circuit from its input-output behavior.

Figure 9 shows the solution process of *NOVEL* in solving a stuck-at-fault problem, ssa7552-038. The eight plots show the number of unsatisfied clauses versus the number of trials (each involving 625 flips for this problem) from different starting points. We note that all runs have found satisfying solutions, and that the number of satisfying clauses is not monotonically decreasing. The latter is caused by changes in the Lagrange variables, which are not shown in this figure. Whenever the trajectory gets in a region with no satisfying solution, the Lagrange variables will bring it out after the trajectory finishes its short descent in the region. This process is repeated in the global search phase until the 'right' region is reached, and the local descent will identify a satisfying assignment in that region.

In Table 4, we compare *NOVEL* with the published results on GSAT. We tested each problem ten times and recorded its minimum, maximum, and average num-

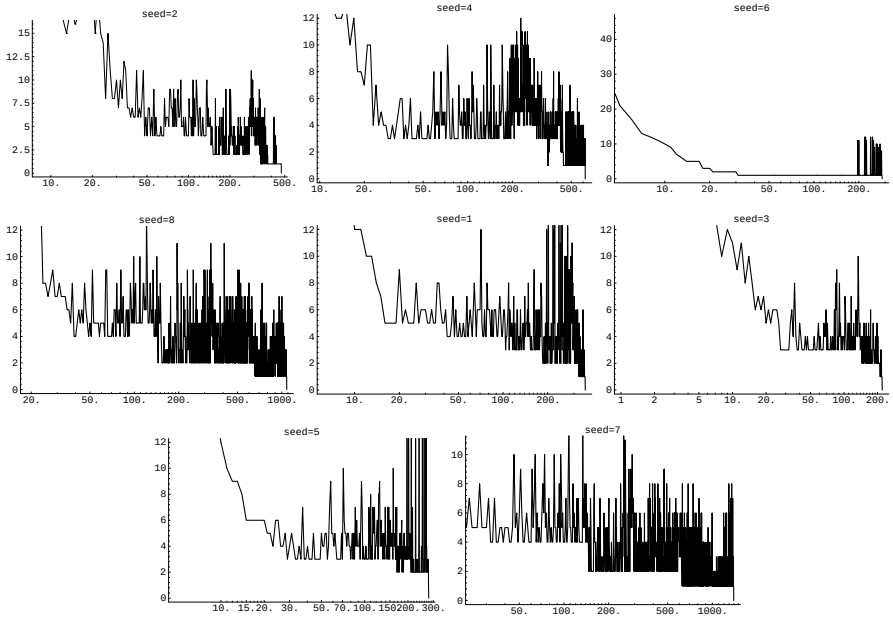

Figure 9. Trajectories of ssa7552-038 showing number of unsatisfied clauses versus number of trials (each involving 625 flips) from different starting points. Other conditions are identical.

Table 5. Results on Boolean Circuit Induction Problems: comparison between *NOVEL*'s results on Sun SS 10/51 and GSAT's published results [75] on SGI Challenge and Kamath *et al.*'s published results on VAX 11/780 [49].

Problem			*NOVEL* (Max-Flips=(5*#vars) per try)							GSAT	Kamath
ID	Var.	Clauses	Tries			Time (sec.)			# Clauses unsat'ed	Time (sec.)	Time (sec.)
			max	min	avg	max	min	avg			
ii16a1	1650	19368	179	16	109	91	9	56	0	274	2039
ii16b1	1728	24792	85	37	61	47	22	34	0	2540	78
ii16c1	1580	16467	8	1	4	12	1	4	0	4	758
ii16d1	1230	15901	157	2	86	59	2	33	0	112	1547
ii16e1	1245	14766	1	1	1	2	1	2	0	2	2156

ber of trials, CPU time, and number of unsatisfied clauses. The results show that our algorithm is comparable to GSAT in terms of average CPU time (as a SGI Challenge is faster than a Sun SS 10/50). Further, *NOVEL* successfully finds satisfying assignments for all the ten trials, using integral seeds 0 thru 9 in the random number generator (*drand48()*) to generate initial assignments.

Table 6. Results on Boolean Circuit Induction Problems: comparison between *NOVEL*'s results on Sun SS 10/51 and Kamath *et al.*'s published results on VAX 11/780 [49]

Problem			*NOVEL* (Max-Flips= (5*#vars) per try)							Kamath
ID	Var.	Clauses	Tries			Time (sec.)			# Clauses unsat'd	Time (sec.)
			max	min	avg	max	min	avg		
ii16a2	1602	23281	136	90	102	63	44	49	0	608
ii16b2	1076	16121	59	22	38	21	8	13	0	236
ii16c2	924	13803	127	70	107	34	20	29	0	521
ii16d2	836	12461	151	71	99	37	18	26	0	544
ii16e2	532	7825	20	1	9	4	1	2	0	376
ii32a1	459	9212	122	2	50	27	1	12	0	177
ii32b1	228	1374	41	7	20	13	3	6	0	5
ii32b2	261	2558	39	7	14	24	5	9	0	57
ii32b3	348	5734	83	2	33	19	5	14	0	190
ii32b4	381	9618	110	1	38	26	1	11	0	259
ii32c1	225	1280	18	5	11	6	2	4	0	24
ii32c2	249	2182	9	6	8	6	4	5	0	9
ii32c3	279	3272	40	14	26	48	16	31	0	14
ii32c4	759	20862	174	58	118	141	32	80	0	155
ii32d1	332	2703	43	14	25	17	6	10	0	66
ii32d2	404	5153	22	7	13	21	6	12	0	178
ii32d3	824	19478	132	3	61	66	2	30	0	1227
ii32e1	222	1186	13	5	9	5	2	3	0	8
ii32e2	267	2746	9	4	7	7	3	5	0	10
ii32e3	330	5020	12	4	9	18	7	13	0	133
ii32e4	387	7106	77	5	54	20	12	14	0	277
ii32e5	522	11636	103	2	91	30	1	24	0	390

In Tables 5 and 6, we compare the performance of *NOVEL*, GSAT, and Kamath *et al.*'s integer programming method in solving the circuit induction benchmarks. We have used only the published timing results of Kamath *et al.* [49] who ran their experiments on a VAX 8700 computer, and Selman's GSAT [75] who ran his experiments on a SGI Challenge computer. We did not attempt to reproduce these results on the same platform, as some runs may depend on the initial conditions used. Our results on *NOVEL* were obtained on a Sun SS10/51.

Tables 5 and 6 show that *NOVEL*'s results are better than previous results on these problems. Our results were run using ten random initial points generated by integral random seeds between 0 and 9. Since our algorithm has less dependence on initial points, all ten runs have found satisfiable assignments. More extensive results on applying this method have been discussed elsewhere [98].

One of the major advantages of *NOVEL* is its ability to escape from local minima without restarts. Consequently, it is able to find feasible assignments irrespective of its initial points. In contrast, descent methods like GSAT have to rely on properly chosen initial assignments. If an initial assignment is not in a promising region, then descent methods cannot find a solution. In such a case, these methods will rely on a good sampling procedure to find new starting points.

7. Conclusions

We have proposed a global optimization method for both continuous and discrete search problems based on a continuous trace that traverses a problem space. *NOVEL* consists of a global search phase and a local search phase. It produces an information-bearing trajectory from which sample points can be identified as starting points for local search. Due to informed decisions in selecting good starting points, *NOVEL* avoids many unnecessary efforts in redetermining already known regions of attraction, and spends more time in finding new unvisited ones. Its complexity is related to the number of "regions of attraction" rather than on the number of dimensions. It is better than multi-start methods that perform blind sampling, as well as random search algorithms (such as genetic algorithms and simulated annealing) that do not exploit gradient information in their global search strategies. It incurs less overhead as compared to generalized gradient methods and other deterministic global optimization methods. It further can be integrated into any existing local search method without affecting its global search.

We have successfully applied *NOVEL* to solve a number of important benchmark problems derived from manufacturing, computed aided design, and others. It has better performance as compared to existing methods for most of these problems and has similar performance for others.

Our future work in this area will be on finding better trace functions, parallelizing the execution on massively parallel computers, and studying many challenging applications in neural network learning and signal processing.

Acknowledgments

We would like to thank Bart Selman at AT&T for providing us with his recent results on GSAT [74]. We are also grateful to the many fruitful comments from Yi Shang and Tao Wang at the Center for Reliable and High Performance Computing, University of Illinois at Urbana-Champaign, throughout the development of this project. Special thanks goes to Tao for his thorough proofreading of earlier versions of this manuscript.

Notes

1. *Mathematica* is a software package developed by Wolfram Research, Champaign, Illinois.
2. LSODE, or Livermore Solver for Ordinary Differential Equations, is the basic solver of *ODEPACK* [42] developed in Lawrence Livermore National Laboratory.

References

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

2. E. H. L. Aarts and P. J. van Laarhoven. *Simulated Annealing: Theory and Practice*. Wiley, New York, 1987.
3. F. Aluffi-Pentini, V. Parisi, and F. Zirilli. Global optimization and stochastic differential equations. *Journal of Optimization Theory and Applications*, 47(1):1–16, September 1985.
4. N. Anderson and G. Walsh. A graphical method for a class of Branin trajectories. *Journal of Optimization Theory and Applications*, 49(3):367–374, June 1986.
5. D. H. Ballard, C. O. Jelinek, and R. Schinzinger. An algorithm for the solution of constrained polynomial programming problems. *The Computer Journal*, 17:261–266, 1974.
6. W. Baritompa. Accelerations for a variety of global optimization methods. *Journal of Global Optimization*, 4:37–45, 1994.
7. W. Baritompa and A. Cutler. Accelerations for global optimization covering methods using second derivatives. *Journal of Global Optimization*, 4:329–341, 1994.
8. B. Betrò and F. Schoen. Sequential stopping rules for the multistart algorithm in global optimization. *Mathematical Programming*, 38, 1987.
9. B. Betrò and F. Schoen. Optimal and sub-optimal stopping rules for the multistart algorithm in global optimization. *Mathematical Programming*, 57:445–458, 1992.
10. C. G. E. Boender and A. H. G. Rinnooy Kan. Bayesian stopping rules for multi-start global optimization methods. *Mathematical Programming*, 36, 1987.
11. C. G. E. Boender and A. H. G. Rinnooy Kan. On when to stop sampling for the maximum. *Journal of Global Optimization*, 1(4):331–340, 1991.
12. C. G. E. Boender, A. H. G. Rinnooy Kan, L. Stougie, and G. T. Timmer. A stochastic method for global optimization. *Mathematical Programming*, 22:125–140, 1982.
13. I. O. Bohachevsky, M. E. Johnson, and M. L. Stein. Generalized simulated annealing for function optimization. *Technometrics*, 28:209–217, 1986.
14. V. Cerny. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45:41–51, 1985.
15. B. C. Cetin, J. Barben, and J. W. Burdick. Terminal repeller unconstrained subenergy tunneling (TRUST) for fast global optimization. *Journal of Optimization Theory and Applications*, 77, April 1993.
16. A. Corana, M. Marchesi, C. Martini, and S. Ridella. Minimizing multimodal functions of continuous variables with the simulated annealing algorithm. *ACM Trans. Math. Software*, 13:2–280, 1987.
17. A. Davenport, E. Tsang, C. Wang, and K. Zhu. Genet: A connectionist architecture for solving constraint satisfaction problems by iterative improvement. In *Proc. of the 12th National Conf. on Artificial Intelligence*, pages 325–330, Seattle, WA, 1994.
18. M. Davis and H. Putnam. A computing procedure for quantification theory. *J. Assoc. Comput. Mach.*, 7:201–215, 1960.
19. I. Diener and R. Schaback. An extended continuous Newton method. *Journal of Optimization Theory and Applications*, 67(1):57–77, October 1990.
20. L. C. W. Dixon and G. P. Szegö, editors. *Towards global optimisation 2*. North-Holland Pub. Co., Amsterdam, The Netherlands., 1978.
21. Y. G. Evtushenko, M. A. Potapov, and V. V. Korotkich. Numerical methods for global optimization. In C. A. Floudas and P. M. Pardalos, editors, *Recent Advances in Global Optimization*, pages 274–297. Princeton University Press, 1992.
22. C. Fleurent and J. A. Ferland. Object-oriented implementation of heuristic search methods for graph coloring, maximum clique, and satisfiability. In M. A. Trick and D. S. Johnson, editors, *The second DIMACS Challenge special issue*, 1994.
23. C. A. Floudas and P. M. Pardalos. *A Collection of Test Problems for Constrained Global Optimization Algorithms*, volume 455 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990.
24. C. A. Floudas and P. M. Pardalos, editors. *Recent Advances in GLobal Optimization*. Princeton University Press, 1992.
25. D. B. Fogel. An introduction to simulated evolutionary optimization. *IEEE Trans. Neural Networks*, 5(1):3–14, January 1994.

26. R. P. Ge and Y. F. Qin. A class of filled functions for finding global minimizers of a function of several variables. *Journal of Optimization Theory and Applications*, 54(2):241–252, 1987.
27. I. Gent and T. Walsh. Towards an understanding of hill-climbing procedures for SAT. In *Proc. of the 11th National Conf. on Artificial Intelligence*, pages 28–33, Washington, DC, 1993.
28. A. O. Griewank. Generalized descent for global optimization. *Journal of Optimization Theory and Applications*, 34:11–39, 1981.
29. J. Gu. *Parallel Algorithms and Architectures for Very Fast AI Search.* PhD thesis, Dept. of Computer Science, University of Utah, August 1989.
30. J. Gu. How to solve very large-scale satisfiability (VLSS) problems. Technical Report UCECE-TR-90-002, Univ. of Calgary, Canada, October 1990.
31. J. Gu. Efficient local search for very large-scale satisfiability problems. *SIGART Bulletin*, 3(1):8–12, January 1992.
32. J. Gu. On optimizing a search problem. In N. G. Bourbakis, editor, *Artificial Intelligence Methods and Applications.* World Scientific Publishers, 1992.
33. J. Gu. The UniSAT problem models (appendix). *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(8):865, Aug 1992.
34. J. Gu. Local search for satisfiability (SAT) problems. *IEEE Trans. on Systems, Man, and Cybernetics*, 23(4):1108–1129, 1993.
35. J. Gu. Global optimization for satisfiability (SAT) problems. *IEEE Trans. on Knowledge and Data Engineering*, 6(3):361–381, Jun 1994.
36. J. Gu and Q.-P. Gu. Average time complexities of several local search algorithms for the satisfiability problem (sat). Technical Report UCECE-TR-91-004, Univ. of Calgary, Canada, 1991.
37. J. Gu and Q.-P. Gu. Average time complexity of the sat1.3 algorithm. Technical report, Tech. Rep., Univ. of Calgary, Canada, 1992.
38. J. Gu and W. Wang. A novel discrete relaxation architecture. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(8):857–865, August 1992.
39. E. Hansen. Bounding the set of solutions of a perturbed global optimization problem. *Journal of Global Optimization*, 1(4):359–374, 1991.
40. E. R. Hansen. *Global optimization using interval analysis.* M. Dekker, New York, 1992.
41. L He and E. Polak. Multistart method with estimation scheme for global satisfying problems. *Journal of Global Optimization*, 3:139–156, 1993.
42. A. C. Hindmarsh. ODEPACK, a systematized collection of ode solvers. In R. S. Stepleman et al., editor, *scientific computing*, pages 55–64. north-holland, amsterdam, 1983.
43. J. H. Holland. *Adaption in Natural and Adaptive Systems.* University of Michigan Press, Ann Arbor, 1975.
44. J. J. Hopfield and D. W. Tank. Neural computation of decisions in optimization problems. *Biological Cybernetics*, 52:141–152, 1985.
45. R. Horst and H. Tuy. *Global optimization: Deterministic approaches.* Springer-Verlag, Berlin, 1993.
46. Reiner Horst and P. M. Pardalos, editors. *Handbook of Global Optimization.* Kluwer Academic Publishers, 1994.
47. L. Ingber. *Adaptive Simulated Annealing (ASA).* Lester Ingber Research, 1995.
48. A. E. W. Jones and G. W. Forbes. An adaptive simulated annealing algorithm for global optimization over continuous variables. *Journal of Optimization Theory and Applications*, 6:1–37, 1995.
49. A. P. Kamath, N. K. Karmarkar, K. G. Ramakrishnan, and M. G. C. Resende. A continuous approach to inductive inference. *Mathematical Programming*, 57:215–238, 1992.
50. S. Kirkpatrick, Jr. C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.
51. H. J. Kushner. A new method of locating the maximum of an arbitrary multipeak curve in the presence of noise. In *Proc. Joint Automatic Control Conf.*, 1963.
52. R. W. Beckerand G. W. Lago. A global optimization algorithm. In *Proc. of the 8th Allerton Conf. on Circuits and Systems Theory*, pages 3–12, Monticello, Illinois, 1970.

53. S. Lucidi and M. Piccioni. Random tunneling by means of acceptance-rejection sampling for global optimization. *Journal of Optimization Theory and Applications*, 62:255–277, 1989.
54. D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley Publishing Company, 1984.
55. Z. Michalewicz. *Genetic Algorithms + Data Structure = Evolution Programs*. Springer-Verlag, 1994.
56. S. Minton, M. D. Johnson, A. B. Philips, and P. Laird. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58:161–205, Dec 1992.
57. D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of SAT problems. In *Proc. of the 10th National Conf. on Artificial Intelligence*, pages 459–465, 1992.
58. J. Mockus. *Bayesian Approach to Global Optimization*. Kluwer Academic Publishers, Dordrecht-London-Boston, 1989.
59. J. Mockus. Application of bayesian approach to numerical methods of global and stochastic optimization. *Journal of Global Optimization*, 4:347–365, 1994.
60. R. Moore and E. Hansen amd A. Leclerc. Rigorous methods for global optimization. In C. A. Floudas and P. M. Pardalos, editors, *Recent Advances in Global Optimization*, pages 321–342. Princeton University Press, 1992.
61. P. Morris. The breakout method for escaping from local minima. In *Proc. of the 11th National Conf. on Artificial Intelligence*, pages 40–45, Washington, DC, 1993.
62. P. M. Pardalos. *Complexity in numerical optimization*. World Scientific, Singapore and River Edge, N.J., 1993.
63. P. M. Pardalos and J. B. Rosen. *Constrained Global Optimization: Algorithms and Applications*, volume 268 of *Lecture Notes in Computer Science*. Springer-Verlag, 1987.
64. N. R. Patel, R. L. Smith, and Z. B. Zabinsky. Pure adaptive search in Monte Carlo optimization. *Mathematical Programming*, 43:317–328, 1988.
65. M. Piccioni. A combined multistart-annealing algorithm for continuous global optimization. Technical Report 87-45, Systems and Research Center, The University of Maryland, College Park MD, 1987.
66. D. Powell and M. M. Skolnick. Using genetic algorithms in engineering design optimization with nonlinear constraints. In S. Forrest, editor, *Proc. of the Fifth Int'l Conf. on Genetic Algorithms*, pages 424–431. Morgan Kaufmann, 1993.
67. H. E. Romeijn and R. L. Smith. Simulated annealing for constrained global optimization. *Journal of Global Optimization*, 5(2):101–126, September 1994.
68. R. Rotondi. A new method for global optimization based on the k-th nearest neighbor. Technical report, CNR-IAMI, Milano, 1987.
69. M. S. Sarma. On the convergence of the Baba and Dorea random optimization methods. *Journal of Optimization Theory and Applications*, 66:337–343, 1990.
70. S. Schäffler and H. Warsitz. A trajectory-following method for unconstrained optimization. *Journal of Optimization Theory and Applications*, 67(1):133–140, October 1990.
71. I. P. Schagen. Internal modeling of objective functions for global optimization. *Journal of Optimization Theory and Applications*, 51(2), 1986.
72. F. Schoen. Stochastic techniques for global optimization: A survey on recent advances. *Journal of Global Optimization*, 1(3):207–228, 1991.
73. R. Sebastiani. Applying GSAT to non-clausal formulas. *Journal of Artificial Intelligence Research*, 1:309–314, 1994.
74. B. Selman, 1995. private communication.
75. B. Selman and H. Kautz. Domain-independent extensions to GSAT: Solving large structured satisfiability problems. In *Proc. of the 13th Int'l Joint Conf. on Artificial Intelligence*, pages 290–295, 1993.
76. B. Selman, H. Kautz, and B. Cohen. Local search strategies for satisfiability testing. In *Proc. of the Second DIMACS Challenge Workshop on Cliques, Coloring, and Satisfiability, Rutgers University*, pages 290–295, oct 1993.
77. B. Selman, H. Kautz, and B. Cohen. Noise strategies for improving local search. In *Proc. of the 12th National Conf. on Artificial Intelligence*, pages 337–343, Seattle, WA, 1994.

78. B. Selman and H. A. Kautz. An empirical study of greedy local search for satisfiability testing. In *Proc. of the 11th National Conf. on Artificial Intelligence*, pages 46–51, Washington, DC, 1993.
79. B. Selman, H. J. Levesque, and D. G. Mitchell. A new method for solving hard satisfiability problems. In *Proc. of AAAI-92*, pages 440–446, San Jose, CA, 1992.
80. Y. Shang and B. W. Wah. Global optimization for neural network training. *IEEE Computer*, 29:45–54, March 1996.
81. J. A. Snyman and L. P. Fatti. A multi-start global minimization algorithm with dynamic search trajectories. *Journal of Optimization Theory and Applications*, 54(1):121–141, July 1987.
82. R. Socic and J. Gu. Fast search algorithms for the N-queen problem. *IEEE Trans. on Systems, Man, and Cybernetics*, 21(6):1572–1576, November 1991.
83. R. Sosič and J. Gu. A polynomial time algorithm for the n-queens problem. *SIGART Bulletin*, 1(3):7–11, October 1990.
84. R. Sosič and J. Gu. 3,000,000 queens in less than one minute. *SIGART Bulletin*, 2(2):22–24, April 1991.
85. R. Sosič and J. Gu. Efficient local search with conflict minimization: A case study of the n-queens problem. *IEEE Trans. on Knowledge and Data Engineering*, 6(5):661–668, 1994.
86. R. Spaans and R. Luus. Importance of search-domain reduction in random optimization. *Journal of Optimization Theory and Applications*, 75(3):635–638, December 1992.
87. H. S. Stone and J. M. Stone. Efficient search techniques– an empirical study of the n-queens problem. *IBM J. Res. Dev.*, 31:464–474, 1987.
88. B. E. Stuckman. A global search method for optimizing nonlinear systems. *IEEE Trans. on Systems, Man, and Cybernetics*, 18(6):965–977, 1988.
89. E. G. Sturua and S. K. Zavriev. A trajectory algorithm based on the gradient method I. the search on the quasioptimal trajectories. *Journal of Global Optimization*, 1991(4):375–388, 1991.
90. G. T. Timmer. *Global Optimization: A Stochastic Approach.* PhD thesis, Erasmus University, Rotterdam, 1984.
91. A. Törn. Global optimization as a combination of global and local search. *Gothenburg Business Adm. Studies*, 17:191–206, 1973.
92. A. Törn. Cluster analysis using seed points and density determined hyperspheres as an aid to global optimization. *IEEE Trans. Syst. Men and Cybernetics*, 7:610–616, 1977.
93. A. Törn and S. Viitanen. Topographical global optimization. In C. A. Floudas and P. M. Pardalos, editors, *Recent Advances in Global Optimization*, pages 385–398. Princeton University Press, 1992.
94. A. Törn and A. Žilinskas. *Global Optimization.* Springer-Verlag, 1989.
95. D. Vanderbilt and S. G. Louie. A Monte Carlo simulated annealing approach to optimization over continuous variables. *Journal of Computational Physics*, 56:259–271, 1984.
96. T. L. Vincent, B. S. Goh, and K. L. Teo. Trajectory-following algorithms for min-max optimization problems. *Journal of Optimization Theory and Applications*, 75(3):501–519, December 1992.
97. A. Žilinskas. A review of statistical models for global optimization. *Journal of Global Optimization*, 2:145–153, 1992.
98. B. W. Wah and Y. Shang. A discrete lagrangian-based global-search method for solving satisfiability problems. In Ding-Zhu Du, Jun Gu, and Panos Pardalos, editors, *Proc. DIMACS Workshop on Satisfiability Problem: Theory and Applications.* American Mathematical Society, March 1996.
99. B. W. Wah, Y. Shang, T. Wang, and T. Yu. Global optimization design of QMF filter banks. In *Proc. IEEE Midwest Symposium on Circuits and Systems*, Iowa City, Iowa, (accepted to appear) Aug. 1996. IEEE.
100. Z. B. Zabinsky and R. L. Smith. Pure adaptive search in global optimization. *Mathematical Programming*, 53:323–338, 1992.
101. Z. B. Zabinsky, *et al.* Improving hit-and-run for global optimization. *Journal of Global Optimization*, 3:171–192, 1993.

Received Date: July 18, 1995
Accepted Date:
Final Manuscript Date: