Rolling algorithm with multiple runs — a non-linear discrete optimization algorithm

Dewi Rahardja *
Department of Statistical Science
Baylor University
Waco, TX 76798
U.S.A.

Yan D. Zhao †
Eli Lilly and Company
Indianapolis, IN 46285
U.S.A.

Han Wu ‡
Department of Mathematics
Austin Peay State University
P.O. Box 4626
Clarksville, TN 37044
U.S.A.

Abstract
Discrete Optimization (DO) problems are often encountered in real world applications. Many researchers believe that DO problems are more difficult to solve than continuous optimization problems, especially when the problems are nonlinear. In addition, although global search heuristics such as the Simulated Annealing (SA) have been successfully applied to many nonlinear DO problems, it is commonly recognized that no single algorithm works well for all problems. Therefore, new alternative algorithms are desired by researchers. In this paper we propose a simple Rolling Algorithm (RA) for solving DO problems. Because RA is easy to implement and takes little resources, we recommend multiple runs of RA for a particular application. We use simulations to show that the RA can be used to solve both discrete and continuous optimization problems. Finally, we observe in our examples that the performance of RA is comparable to that of the SA.

Keywords and phrases: Heuristics/algorithm, discrete optimization problems, discrete optimization algorithms, rolling algorithm, multiple runs.

*E-mail: Rahardja@gmail.com
†E-mail: zhaoyan1@gmail.com
‡E-mail: wuh@apsu.edu

Journal of Statistics & Management Systems
Vol. 12 (2009), No. 5, pp. 917–925
© Taru Publications
1. Introduction

Discrete Optimization (DO) problems often involve discrete variables and parameter spaces, multi-modal functions, and non-differentiable functions. Such problems can arise in forecasting, dynamic modeling, game theory models, minimization of waste, optimizing resources, scheduling, and so on.

A growing interest has been shown for DO algorithms since continuous optimization algorithms were well established in the 1980s. Reviews of the developments in discrete optimization methods have been presented by Parker and Rardin (1988), Nemhauser et al. (1989), and Arora and Haug (1994). These reviews indicated that DO is more difficult to solve and requires more computational cost than continuous optimization.

Current DO algorithms can be classified as either deterministic or probabilistic. The branch and bound method (BBM) is likely the best known and most frequently used deterministic DO algorithm. It is important to note that the BBM is guaranteed to find the global optimum if the problem is linear or convex, but not guaranteed to converge in the general nonconvex case. The method is based on the sequential analysis of a discrete tree for each variable and the computational cost grows exponentially with the number of variables. Therefore, the method is not suitable for the analysis of problem with many variables.

The three most widely used probabilistic DO algorithms are Genetic Algorithm (GA), Tabu Search (TS), and Simulated Annealing (SA). These algorithms are able to provide feasible global solutions to optimization problems for which calculus-based or gradient-based techniques are infeasible or impossible. For the GA literatures, see Holland (1975, 1992), Shaefer (1985), and Michalewicz (1996). For the TS literatures, see Glover (1977), Glover (1990), Glover and Laguana (1993), and Cvi jovik and Klinowski (1995). The SA was first introduced by Kirkpatrick et al. (1983) and Cerny (1985) and has root in the work of Metropolis et al. (1953). Numerous successful applications of SA can be found in Hajek (1988), Collins et al. (1988) and Belisle (1992).

It is commonly recognized that no practical DO algorithm has emerged as the standard. Therefore, new alternative DO algorithms are much desired by the researchers. In this article, we propose a new probabilistic non-gradient based algorithm – Rolling Algorithm (RA). Because RA is easy to implement and takes little resources, we recommend multiple runs of RA for a particular application and the optimum from the multiple runs is taken as the final solution.
ROLLING ALGORITHM WITH MULTIPLE RUNS

The reasons for multiple runs of RA are as follows. To solve a particular DO problem, the vast majority of the researchers apply a single run of their favorite algorithms. However, for algorithms with random starting values, recently some researchers advocate multiple runs of algorithms to achieve higher efficiency without increasing computing resources. This is especially true for problems with a number of widely separated local optima because a single run has a larger chance of being entraped in a local optimum than multiple independent runs. Moreover, the approach of multiple runs can be further leveraged by utilizing parallel computation. Luke (2001) discovered that for genetic algorithms there is a maximal number of generations beyond which it is irrational to plan a single run; instead it makes more sense to do multiple shorter runs. Wan and Chen (2003) demonstrated that they can always improve the chance of finding a solution by running the same Stochastic Search Algorithm (SSA) multiple times from random starting points. Schutte and Haftka (2005) reported that the multi-run strategy with small Particle Swarm Optimizer (PSO) populations delivers higher global convergence probability than a single run with a large population and an equal number of fitness evaluations.

The rest of the article is organized as follows. In Section 2 we propose the RA with multiple runs. In Section 3 we apply the RA to two example cases and compare its performance with the SA using the Mean Best Objective Function Values (MBOFV) method detailed in Rahardja et al. (2007). Some general conclusion is presented in Section 4.

2. Methods — rolling algorithm

We formulate the general optimization problem as follows:

\[
\begin{align*}
\min_{\mathbf{x}} & \quad f(\mathbf{x}) \\
\text{s.t.} & \quad g_s(\mathbf{x}) \leq 0, \quad s = 1, \ldots, n \\
& \quad x_i \in A_i = \{a_{i,1}, \ldots, a_{i,m_i}\}, \ i = 1, \ldots, k
\end{align*}
\]

where \( f \) and \( g_s \)'s are real functions of \( k \) variables, \( n \) indicates the number of inequality constraints, \( A_i \) is the candidate subset and \( m_i \) is the number of candidates for the \( i \)th variable.

In order to find a solution \( \mathbf{x}^* = (x_1^*, \ldots, x_k^*) \) to Equation (1), we propose a new probabilistic algorithm which we name as the Rolling Algorithm with \( r \) rolling variables \( \text{RA}(r) \). The \( \text{RA}(r) \) first uses a random permutation \( O_1, \ldots, O_k \) of \( 1, \ldots, k \) as the order of rolling variables. For
Start with a random choice of \( k - r \) values \( x_{O_1}, \ldots, x_{O_{k-r}} \).

(2) Hold \( x_{O_1}, \ldots, x_{O_{k-r}} \) fixed and use direct enumeration to find \( x_{O_{k-r+1}}, \ldots, x_{O_k}^* \) so that \( x_{O_1}, \ldots, x_{O_{k-r}}, x_{O_{k-r+1}}^*, \ldots, x_{O_k}^* \) minimizes \( f \) subject to all the inequality and side constraints in Equation (1). This step performs minimization by rolling over the last \( r \) variables in the order of \( O_1, \ldots, O_k \) while holding the first \( k - r \) variables fixed.

(3) Hold \( x_{O_1}, \ldots, x_{O_{k-2}}, x_{O_{k-r+1}}^*, \ldots, x_{O_k}^* \) fixed and use direct enumeration to find \( x_{O_{k-3}}, \ldots, x_{O_{k-r+1}}^*, \ldots, x_{O_k}^* \) so that \( x_{O_1}, \ldots, x_{O_{k-2}}, x_{O_{k-3}}^*, \ldots, x_{O_{k-r+1}}^*, \ldots, x_{O_k}^* \) minimizes \( f \) subject to all the inequality and side constraints in Equation (1). This step performs minimization by rolling over the second-to-last \( r \) variables in the order of \( O_1, \ldots, O_k \) while holding the other variables fixed.

(4) Continue until finally, find integers \( c \) and \( d \) so that \( k = cr + d \) and \( d < r \). Hold \( x_{O_{k-d+1}}, \ldots, x_{O_k}^* \) fixed and use direct enumeration to find \( x_{O_1}, \ldots, x_{O_{k-d}}, x_{O_k}^* \) so that \( x_{O_1}, \ldots, x_{O_{k-d}}, x_{O_{k-d+1}}^*, \ldots, x_{O_k}^* \) minimizes \( f \) subject to all the inequality and side constraints in Equation (1). This step performs minimization by rolling over the first \( d \) variables in the order of \( O_1, \ldots, O_k \) while holding the other variables fixed.

(5) The final solution \( x_{O_1}^*, \ldots, x_{O_k}^* \) and the corresponding objective function value \( f \) are recorded.

Clearly RA\((r)\) is easy to implement and requires little resources when \( r \) is small. For example, consider a DO problem without inequality constraint, RA\((1)\) requires \( \Sigma m_i \) computations of the objective function, as compared with \( \Pi m_i \) computations of the objective function required by the direct computation. The computational cost increases as \( r \) increases and RA\((k)\) is the direct enumeration.

Although 1 run of the RA\((r)\) may produce a local minimum instead of a global minimum for small \( r \), it requires affordable computation of objective function values compared with the direct enumeration as evidenced earlier. Consequently, we will perform multiple runs of RA\((r)\) for a particular problem and use the optimum from the multiple runs as the final solution. Combining RA\((r)\) with multiple runs will increase the convergence probability. For instance, for a single run of RA for a DO problem without inequality constraint, the convergence probability \( p_1 \), has a lower bound \( \Sigma m_i / \Pi m_i \). Although this lower bound may not be
large, the theoretical convergence probability for multiple \((t)\) runs of RA is \(p_t = 1 - (1 - p_1)^t\) which goes to 1 much more rapid than 1 run of the RA.

The choice of number of rolling variables \(r\) and the number of runs \(t\) should be tailored to a particular problem. The researchers should try a range of \(r\) and \(t\) values to see which combination works the best. Generally, one should choose \(r\) and \(t\) so that the total computation cost is within the budget.

3. Example cases

In this section we consider two optimization problems. The first one is a constrained discrete optimization problem: \textit{Traveling Salesman Problem} (TSP), and the second one is a continuous optimization problem: 2-D Rosenbrock function minimization. We apply the RA to solve both problems. Moreover, because the SA (Kirkpatrick et al., 1983) is well established for solving both the TSP and the Rosenbrock problem, we also applied SA to solve both problems.

For the RA algorithms, the orders of rolling variables are randomly chosen at each run. The starting values for both the RA and the SA at each run are also randomly generated. For the SA, the initial temperature (temp) is set at 2000 and the number of function evaluations (tmax) at each temperature is set at 10. Temperatures are decreased according to the logarithmic cooling schedule as given in Belisle (1992, p. 890); specifically, the temperature at iteration \(t\) is set to \(\text{temp}/\log(((t - 1)/\%)/\%\text{tmax}) \ast \text{tmax} + \exp(1))\), where the symbol “%/%” represents the integer modulation operator. A total of 300 maximum iterations are used as a stopping rule. For the TSP, the Gaussian Markov kernel with scale proportional to the actual temperature is used to generate the new candidate point. For the 2-D Rosenbrock function minimization, the new candidate point is generated by swapping two randomly selected cities from the current point.

We use the MBOFV method to compare the performance of multiple runs of RA with the SA. The MBOFV method was developed to assess the convergence performance of multiple runs of an optimization algorithm with random starting values. The following is an outline of how to apply this method:

1. A large number \((N)\) of runs of an algorithm is performed and the objective function value from each run is recorded.
An empirical distribution function (EDF) is computed based on the \( N \) objective function values.

(3) The mean best objective function values (MBOFV) are computed for each Run \( j \), \( j = 1, \ldots, J \).

(4) Plot the MBOFV versus the number of runs \( j \).

3.1 Traveling Salesman Problem (TSP)

For illustration, we consider a 10-city TSP and aim to find the shortest round-trip route that visits each city exactly once and then returns to the starting city. The 10 cities are Athens, Barcelona, Brussels, Calais, Cherbourg, Cologne, Copenhagen, Geneva, Gibraltar, and Hamburg. We apply RA(5), RA(7), and SA to solve this problem.

In order to use the MBOFV method to compare the performance of three algorithms, we first performed 100 initial runs. Then, we plot the mean best objective function values versus the number of runs. From Figure 1 we observe that the performance at each run is in the order of RA(7), SA, RA(5) with RA(7) being the best. It should be noted the difference between the two algorithms RA(7) and SA is not big because even the shortest distance between two cities is more than 200 miles. Therefore, RA(7) and SA give comparable results for the TSP.

3.2 2-D Rosenbrock function minimization

Because real numbers are represented as rational numbers with limited decimal points in modern computers, continuous optimization problems are intrinsically discrete if solved using computer software packages. Therefore, although RA is developed for discrete optimization problems, in this section we attempt to solve continuous optimization problems using RA with multiple runs.

We consider the 2-D Rosenbrock function introduced in Rosenbrock (1960) as a test problem for minimization algorithms:

\[
f(x, y) = (1 - x)^2 + 100(y - x^2)^2
\]

which has a global minimum at \((x, y) = (1, 1)\) where \(f(x, y) = 0\). Due to a long narrow valley present in this function, gradient-based techniques may require a large number of iterations before the solution is found. Therefore, we use the SA as a competitor of RA to minimize the Rosenbrock function. In order to apply RA, we first divide the interval \((0, 2)\) into 20 equally spaced intervals; namely, let \(A_1 = \{0, 0.1, \ldots, 1.9, 2\}\). Then,
we apply the RA(1) to minimize the Rosenbrock function over the grid generated by the Cartesian product of $A_1$ and $A_1$.

**Figure 1**

*Mean Best Objective Function Values (MBOFV) versus number of runs for Traveling Salesman Problem with RA(5), SA, and RA(7)*

In order to use the MBOFV method to compare the performance of three algorithms, we first performed 1,000 initial runs. Then, we plotted the mean best objective function values versus the number of runs for both algorithms. The left and right panels of Figure 2 plotted the first 10 runs and the Run 10 to Run 100, respectively. We see that although SA is better than RA(1) initially, RA(1) begins to outperform SA after about Run 38.

**Figure 2**

*Mean Best Objective Function Values (MBOFV) versus number of runs for 2-D Rosenbrock Minimization with RA(1) and SA*
4. Conclusion

In this paper we propose a new simple discrete nonlinear optimization algorithm the Rolling Algorithm (RA). In addition, we show that the performance of RA can be enhanced by running multiple times. The RA was shown to be a competitive algorithm for a classical discrete optimization problem – the TSP. Finally, we illustrated how to apply the RA to a continuous nonlinear discrete optimization problem and demonstrated its good performance.

References


Received June, 2008