World Academy of Science, Engineering and Technology
International Journal of Mathematical and Computational Sciences
Vol:11, No:5, 2017

Efficiency of Robust Heuristic Gradient Based Enumerative and Tunneling Algorithms for Constrained Integer Programming Problems

Vijaya K. Srivastava, Davide Spinello

Abstract**—**This paper presents performance of two robust gradient-based heuristic optimization procedures based on 3^n enumeration and tunneling approach to seek global optimum of constrained integer problems. Both these procedures consist of two distinct phases for locating the global optimum of integer problems with a linear or non-linear objective function subject to linear or non-linear constraints. In both procedures, in the first phase, a local minimum of the function is found using the gradient approach coupled with hemstitching moves when a constraint is violated in order to return the search to the feasible region. In the second phase, in one optimization procedure, the second sub-procedure examines 3^n integer combinations on the boundary and within hypercube volume encompassing the result neighboring the result from the first phase and in the second optimization procedure a tunneling function is constructed at the local minimum of the first phase so as to find another point on the other side of the barrier where the function value is approximately the same. In the next cycle, the search for the global optimum commences in both optimization procedures again using this new-found point as the starting vector. The search continues and repeated for various step sizes along the function gradient as well as that along the vector normal to the violated constraints until no improvement in optimum value is found. The results from both these proposed optimization methods are presented and compared with one provided by popular MS Excel solver that is provided within MS Office suite and other published results.

Keywords**—**Constrained integer problems, enumerative search algorithm, Heuristic algorithm, tunneling algorithm.

I. Introduction

INTE-GER programming is one of the most interesting and one of the most difficult research areas in mathematical programming and operations research. During the past years, much work has been devoted to the development of algorithms for solving integer problems. Several solution methods include branch and bound, Gomory cutting method are described in [1]. The generalized reduced gradient algorithm used in MS solver for solving nonlinear integer programming problems was developed by Wilde and Beightler [2], and the method implemented by Lasdon et al. is elaborated in report [3]. The use and development of heuristic-based optimization techniques have significantly grown, since they use a population of solutions in their search, it is thus more likely to find the global solution of a given problem [4], [5]. A hybrid algorithm is developed [6] which combines integer programming branch and bound techniques with a genetic algorithm (GA) to gain advantage of the two approaches, i.e. tree search and GA algorithm to provide heuristic search for the optimum solution. Recently, an efficient robust and hybrid heuristic algorithm has been introduced by combining two heuristic optimization techniques, particle swarm optimization [7] and GAs [8].

The methods for solving constrained integer programming reported by the authors in [4], [5] involved an iterative function minimization approach consisting of two phases. In the first phase, a local minimum is found using a steepest descent approach along the function gradient in the feasible region, and if a constraint is violated, the return to a point $\boldsymbol{X_k}$ in the feasible region is carried out using a hemstitching approach [12]. This hemstitching approach uses small moves normal to the violated constraint boundary into the feasible region. During this phase, all moves are restricted to integer points obtained by the discretization of each variable of the *n tuple* design vector. The search for the optimum point after rebounding from a constraint resumes along a direction that is the vector sum [13] of the function and the violated constraint gradients at $\boldsymbol{X_k}$. The optimum point thus found is then used as a starting vector for the next iterative cycle. This process continues until no improvement in the minimum value is possible during this phase. The first phase would generally locate the local minimum within the same valley of the starting vector. This is a general problem of numerical search approaches when dealing with continuous or integer domains. Several well documented approaches have been reported in the open literature to increase the likelihood of capturing a global minimum. However, this problem is further aggravated by the increase in number of local minimums due to the discretization process. In order to circumvent the problem of getting stuck in local minima, the second phase is implemented. In this phase, a 3^n enumeration [4] is carried out using the local optimum found in the first phase as a basis vector. The minimum function value found using the 3^n enumeration is used as a starting vector for the next iterative search. The two-phase cycle continues until there is no further improvement in the optimum result, and the best possible optimum, $\boldsymbol{X}^*$ is attained**.**

While the approach reported in [4] was proven to be very effective in dealing with nonlinear integer optimization problems, as *n* increases the computational time of the above mentioned method increases. For very large values of *n*, the

Vijaya K. Srivastava is with the Department of Mechanical Engineering, University of Ottawa, Ottawa, Canada (e-mail: vsrivast@uottawa.ca).
Davide Spinello is with the Department of Mechanical Engineering, University of Ottawa, Ottawa, Canada.

computational time tends to be prohibitively large for practical purposes. Computation time analysis of the two phases revealed that as n increases, the time spent solving the 3^n enumeration increases exponentially.

In order to circumvent the deficiency of dealing with problems where n is large as well as to increase the robustness of the algorithm by replacing the 3^n enumeration phase, the second phase, with a tunneling approach that was first reported in [9]. This approach is modified to suit the requirements of second optimization procedure reported in [5] and involves creating a pole at the local minimum and constructing a tunneling function whose solution is found using a descent along the gradient of the tunneling function. The solution to the minimization of the tunneling function, $\boldsymbol{x} \neq \boldsymbol{x}^*$, is taken as the starting point $\boldsymbol{x}^0$ for the first phase of the next iterative cycle, and the search for the global optimum continues. The iterative process is stopped when the optimal step size becomes smaller than a pre-assigned value during the first phase or when the search returns to a previously found minimum. During the iterative search for the desired constrained global minimum, the records for all local minima found during each cycle are kept, and the search is terminated when no improvement in function value is found or after a set number of iterations. The minimum of the recorded set is taken as the optimum. Therefore the search algorithm generates a non-increasing sequence $f_1 \geq f_2 \geq : : : \_ \geq f_k$, where the lowest function value f_k, is the global minimum f^*.

MS Excel solver also provides an option to use the Simplex procedure for Linear programming problems [10], [11] and Generalized Reduced Gradient (GRG2) Algorithm for optimizing nonlinear problems [3]. Additionally, several commercial solvers for linear and non-linear integer programming problems are available. Examples include LINGO/LINDO1 commercial software packages and BARON, a software computational tool that utilizes the branch and bound approach developed in [1]-[11] to solve non convex integer programming problems. The optimum solution vector found for various integer programming test problems and computation times of both optimization procedures [4][5] are compared with that provided by MS Excel solver.

A. Problem Statement

The generic minimization problem is stated as follows:

Find $\boldsymbol{X}^*$ that minimizes $f(\boldsymbol{X})$

Subject to:

$$g_j(\boldsymbol{X}) \leq b_j \; j = 1,2,\ldots\ldots \text{m, where, } X_L \leq X \leq X_H \qquad (1)$$

$X \in I^n$, I^n is the set of integer points in $R^n \rightarrow R$. X, X_L, and X_H are the design vector with dimension n, and the lower, and upper bounds on X, respectively.

It is to be noted that satisfying the upper and lower bounds on the design vector X does not imply satisfying the functional constraints $g_j(X)$. A feasible solution X^* is optimal if $f(X^*) \leq f(X)$ for every feasible solution X.

II. Description of the Algorithm

The proposed algorithm is composed of a sequence of cycles, each consists of two phases; a common minimization phase for both heuristic methods [4], [5] followed by and a 3^n enumeration phase for the first heuristic method and a tunneling phase for the second heuristic method. The minimization phase is designed to locate the local minimum in the valley containing the starting point, whether unconstrained or constrained. The descent towards the minimum in the unconstrained domain is carried out along the function gradient,

$$X_{k+1} = X_k + \lambda_k (\nabla f(X_k) / \| (\nabla f(X_k) \|$$

that is converted to nearest integer and the function value is calculated. The step size is increased if the function value decreases or does not change and is decreased otherwise. An optimized step size based on a quadratic approach is described in [4]. However, when the constraints are violated, the search is modified to move along a direction vector $\boldsymbol{Q}$ that is the normalized sum of gradients of the p violated constraints $g_1(X)$, ….. $g_p(X)$ [12]. The i^{th} element of the vector $\boldsymbol{Q}$ is given by:

$$q_i = - \Sigma^p_{k=1} \, \ell_k \, (\eth g_k / \eth x_i) / \sqrt{\Sigma^n_{j=1} (\eth g_k / \eth x_j)^2} \qquad (2)$$

p is the number of violated constraints. The coefficients ℓ_k have values proportional to the magnitude by which the constraints are violated and are defined as follows:

$$\ell_k = (g_k - b_k) / (\Sigma^p_{j=1} (g_j - b_j)$$

The move towards the feasible region is carried out vector along $\boldsymbol{Q}$ as:

$$\mathbf{X}_{j+1} = \mathbf{X}_j + \mathbf{Q} \, (\lambda_o + j\ell) \, , j = 0,1,\ldots,k \qquad (3)$$

$\mathbf{X}_o$ is the vector in the infeasible region, and after k^{th} moves along the director vectors Q that are evaluated at each move j, and λ_o, is the initial step size along the normal to the violated constraint which during each iteration is increased by small amount $\ell \leq 1$ until feasible vector $\mathbf{X}_k$ is realized.

When the feasible region is reached then the search resumes along a direction $\boldsymbol{P}$ that is the vector sum of the normalized function and the violated constraints gradients at the new recovery point in the feasible region [13] as:

$$P = p / \| p \| \qquad (4)$$

where p is given by

$$p = \frac{\nabla f}{\| \nabla f \|} + \sum_{j=1}^{m} \frac{\nabla g_j}{\| \nabla g_j \|}$$

The function minimization along P is carried out using the optimized step size approach reported in [4]. If the search strays again into the infeasible region, the recovery into the

feasible region is carried out using the hemstitching approach as described above.

This phase, the first phase, of the search is terminated if no further improvement in the function value is achieved, or if a previously encountered minimum point is detected. The suboptimum vector is selected as a basis vector of the second procedure where the minimum is selected from 3^n integer combinations vectors neighboring the result from the first procedure. Again here, an iterative cycle ensues until two cycles yield the same answer. This is taken as the desired optimum solution of the problem. The desired optimum solution of the problem can be verified by repeating the search for various starting values and initial step sizes.

In the second heuristic optimization procedure [5] based on tunneling during, the second phase however uses the local minimum found in the phase I, X^*, to calculate a good starting point through the tunnel to the next valley. A tunneling function [9] is defined as follows:

$$T(X) = \frac{f(X) - f(X^*)}{[(X - X^*)^T (X - X^*)]^\eta} \qquad \eta \geq 1$$

For the current application η is set to 1. The root (i.e. zero) of the tunneling function, X^0, is in essence a point at the same plateau in the neighboring valley. The root can be obtained through a classical root finding algorithm, or as implemented in the proposed algorithm the pseudo root is found through a minimization along the gradient of *T(X)*. The function is assumed in this case to be unconstrained, and no discretization of the variables and no hemstitching are performed if the descent strays in the infeasible region. The descent along the tunneling function is terminated when $\left| T(X^*) - T(X^0) \right| \leq \varepsilon$, where ε is a small parameter. The resulting vector, X^0, of the tunneling process is discretized and used as the starting vector for the first phase of the next cycle.

The initial step size λ_o^k for the k^{th} cycle is adjusted as:

$$\lambda_o^{(k+1)} = (1/\Delta)\,\lambda_o^{(k)} \text{ if } f(x^*_k) \geq f(x^*_{k-1})$$
or
$$\lambda_o^{(k+1)} = \Delta\,\lambda_o^{(k)} \text{ if } f(x^*_k) < f(x^*_{k-1}) \qquad (5)$$

where x^*_k is the local minimum and Δ is a multiplier (Δ >1)

During the iterative search for the desired constrained global minimum, the records for all local minima found during each cycle are kept, and the search is terminated when no improvement in function value is found or after a set number of iterations. The minimum of the recorded set is taken as the optimum.

III. Numerical Examples

The algorithm is coded in FORTRAN programming language using double precision real numbers and a 64 bit Watcom FORTRAN compiler was used to compile the main source optimization program linked to the test function routine to create executable file. The coefficients of the test problem and the various parameters used by the optimization algorithm such as step size, acceleration factors etc., are entered in a data file that is read prior to the running of the algorithm. Constrained integer test problems shown in [5] have been solved using both optimization methods [4], [5] and MS Excel solver. Table I summarizes the results of the test problems of [5] as well as the results of these test problems solved by using MS solvers. Additionally, four published complex linear and non-linear constrained optimization test problems including a large scale 40 variables linear programming problem have been solved that demonstrates the usefulness of tunneling algorithm [5]. The results of both heuristic algorithms reported in [4] are compared to the optimum provided by MS solver. These additional test functions provided an opportunity to improve upon in the search for global optimum for heuristic involving tunneling as noted below for solving test problem #10 through # 13.

Test Problem #10 Linear Integer Problem [22]

Minimize $f(x) = - x_3 - x_4 - x_5$

Subject to:

$$20x_1 + 30\,x_2 + x_3 + 2\,x_4 + 2\,x_5 \leq 180$$

$$30x_1 + 20\,x_2 + 2x_3 + x_4 + 2\,x_5 \leq 150$$

$$-60x_1 + x_3 \leq 0,\ -75\,x_2 + x_4 \leq 0$$

$$0 \leq x_i \leq 1\ ,\ i = 1,2\ ;\ 0 \leq x_i \leq 75,\ i = 3,4,5:\ x_i\ \varepsilon\ \mathbb{Z},\ i = 1, 2, \ldots,5$$

A starting vector (1,1,….,1) for 3^n enumerative algorithm [4] provides a local optimum (1,1,40,20,0) with function value -60. An examination of the coefficient of x_3 and x_4 of the first two constraints reveals that a vector (1,1,20,40,0) also with function value -60 is also acceptable. Using this vector as starting vector yields an improved optimum vector (1,1,28,44,0) with function value -72. The perturbation of this solution vector (1,1,28,44,0) will lead to realization of the global optimum (1,1,24,52,0) with minimum function value -76 as reported [22]. The heuristic tunneling procedure [5] with starting vector (1,1,1,1,1) however yields optimum vector (1,1,23,53,0) with function -76, and the reported solution vector (1,1,24,52,0) with function value -76 implies presence of a bi-modal global optimum. MS solver provides reported solution vector (1, 1, 24, 52, 0).

Test Problem #11 Non-linear Integer Problem [22]

Minimize $f(x) = x_1x_2x_3 + x_1x_4x_5 + x_2x_4x_6 + x_6x_7x_8 + x_2x_5x_7$

Subject to :

$$2x_1 + 2\,x_4 + 8x_8 \geq 12$$
$$11x_1 + 7\,x_4 + 13x_6 \geq 41$$
$$6x_2 + 9x_4\,x_6 + 5x_7 \geq 60$$
$$3x_2 + 5x_5 + 7x_8 \geq 42$$
$$9x_3 + 6x_2\,x_7 + 5x_5 \geq 53$$
$$x_5 + 4x_3\,x_7 \geq 13$$
$$2x_1 + 4x_2 + 7x_4 + 3x_5 + x_7 \leq 69$$
$$9x_1\,x_8 + 6x_3\,x_5 + 4x_3\,x_7 \leq 47$$
$$12x_2 + 8x_2\,x_8 + 2x_3\,x_6 \leq 73$$
$$x_3 + 4x_5 + 2x_6 + 9x_8 \leq 31$$

$$0 \le x_i \le 7,\ i = 1, 3, 4, 6, 8;\ 0 \le x_i \le 15,\ i = 2,5,7$$
$$x_i \in \mathbb{Z},\ i = 1, 2, \ldots\ldots,8$$

As observed in Test problem #7 in [5], the presence of types of resource constraints where the equations of resource equations are ≥ or ≤ than a specified quantity then the lower bounds of the variables needs to be carefully selected in order to prevent search from getting trapped in the infeasible region. The lower bound of this test problem was selected as (1,1,1, 1,0,0,0,0), and the upper bound governed by specification of the problem that is shown here as (7,15,7,7,15,7,15,7). The initial starting vector (5,5,1,1,4,1,1,1) that lies in the infeasible region provides the reported global optimum (5,4,1,1,6,3,2,0) with function value 110 using 3^n enumerative algorithm [4] and provides a local optimum (4,4,1,2,6,2,2,0) using tunneling algorithm [5]. MS solver provides a sub-optimum solution vector (2,4,1,4,6,1,2,0) with function value 120, however, with initial starting value (4,4,1,1,11,2,1,0) the referenced global optimum value is obtained, and the search requires a higher execution time.

Test Problem #12 Non-linear Integer Problem with Linear Constraints [23]

Minimize $f(x) = x_1^2 + x_2^2 + 3x_3^2 + 4x_4^2 + 2x_5^2 - 8x_1 - 2x_2 - 3x_3 - x_4 - 2x_5$

Subject to

$$x_1 + 2x_2 + 2x_3 + x_4 + 6x_5 \le 800$$
$$2x_1 + x_2 + 6x_3 \le 200$$
$$x_3 + x_4 + 5x_5 \le 200$$
$$x_1 + x_2 + x_3 + x_4 \ge 48$$
$$x_2 + x_4 + x_5 \ge 34$$
$$6x_1 + 7x_5 \ge 104$$
$$55 \le x_1 + x_2 + x_3 + x_4 + x_5 \le 400$$
$$0 \le x_i \le 99\ ,\ x_i : \text{integer},\ i = 1,2,3,4,5.$$

It is noticed from the 4th and 7th constraint that lower bound of x_5 is 7, so the lower bound of design vector is selected as (0,0,0,0,7), upper bound as specified and a starting vector (20,20,10,20,7) in the infeasible region is selected. The global minimum (16, 22, 5, 5, 7) and the minimum function value 807 is found using heuristic search [4] using 3^n enumeration. In case of heuristic search involving tunneling [5], the multiplier ℓ_i used to adjust the step size of move along orthogonal to the violated constraint played a pivotal role in reaching at the desired point in the feasible from where descent to global minimum was achieved. MS Excel solver also provided the global optimum (16, 22, 5, 5, 7) using the same initial vector.

Test Problem #13 Large Variable Integer Problem with Linear Constraints [24]

$$\text{Maximize } f(x) = 215x_1+116x_2+670x_3+924x_4+510x_5+600x_6+424x_7+942x_8+43x_9+369x_{10}+408x_{11}+52x_{12}+319x_{13}+214x_{14}+851x_{15}+394x_{16}+88x_{17}+124x_{18}+17x_{19}+779x_{20}+278x_{21}+258x_{22}+271x_{23}+281x_{24}+326x_{25}+819x_{26}+485x_{27}+454x_{28}+297x_{29}+53x_{30}+136x_{31}+796x_{32}+114x_{33}+43x_{34}+80x_{35}+268x_{36}+179x_{37}+8x_{38}+105x_{39}+281x_{40}$$

Subject to:

$$9x_1+11x_2+6x_3+x_4+7x_5+9x_6+10x_7+3x_8+11x_9+11x_{10}+2x_{11}+x_{12}+16x_{13}+18x_{14}+2x_{15}+x_{16}+x_{17}+2x_{18}+3x_{19}+4x_{20}+7x_{21}+6x_{22}+2x_{23}+2x_{24}+x_{25}+2x_{26}+x_{27}+8x_{28}+10x_{29}+2x_{30}+x_{31}+9x_{32}+x_{33}+9x_{34}+2x_{35}+4x_{36}+10x_{37}+8x_{38}+6x_{39}+x_{40} \le 25{,}000$$

$$5x_1+3x_2+2x_3+7x_4+7x_5+3x_6+6x_7+2x_8+15x_9+8x_{10}+16x_{11}+x_{12}+2x_{13}+2x_{14}+7x_{15}+7x_{16}+2x_{17}+2x_{18}+4x_{19}+3x_{20}+2x_{21}+13x_{22}+8x_{23}+2x_{24}+3x_{25}+4x_{26}+3x_{27}+2x_{28}+x_{29}+10x_{30}+6x_{31}+3x_{32}+4x_{33}+x_{34}+8x_{35}+6x_{36}+3x_{37}+4x_{38}+6x_{39}+2x_{40} \le 25{,}000$$

$$3x_1+4x_2+6x_3+2x_4+2x_5+3x_6+7x_7+10x_8+3x_9+7x_{10}+2x_{11}+16x_{12}+3x_{13}+3x_{14}+9x_{15}+8x_{16}+9x_{17}+7x_{18}+6x_{19}+16x_{20}+12x_{21}+x_{22}+3x_{23}+14x_{24}+7x_{25}+13x_{26}+6x_{27}+16x_{28}+3x_{29}+2x_{30}+x_{31}+2x_{32}+8x_{33}+3x_{34}+2x_{35}+7x_{36}+x_{37}+2x_{38}+6x_{39}+5x_{40} \le 25{,}000$$

$$10 \le x_i \le 99\ ,\ (i = 1,2,\ldots,20)\quad 20 \le x_i \le 99\ i = 21,22,\ldots..,40)\ x_i : \text{integer},$$

The problem was solved using tunneling algorithm [5] and the solution vector provided (99,99,……..,99) with function 1,352,439 that agrees with the solution provided by MS solver. This result is superior to the one reported in [24] as $f(x^*) = 1{,}030{,}361$, and the solution vector is shown in Table I.

IV. Discussion

This paper presents the comparison of two robust heuristic methods [4], [5] developed by the authors for dealing with the optimization of general class of integer problems with a linear or non-linear objective function subject to linear or non-linear constraints and presented the effectiveness of these search algorithms for finding global optimum by comparison with reported results and those provided by MS Excel solver, a commonly available commercially tool within MS Office suite. The method generally does not require the user to guess the starting vector or the parameters required for the operation of the method, like step size or accelerations factors. However, for problems involving complex constraints, the lower boundary is suitably selected to avoid the search from getting trapped in infeasible region. In case for test problem #12, the choice of the multiplier used to adjust the step size during the hemstitching for the returning the search to feasible region played pivotal role in returning the search to appropriate region of the feasible region from where the descent to global optimum was achieved. Through the use of a tunneling algorithm, higher dimensionality problems such as #8 and #13, can be solved, and the method seeks its own starting vector for its different iterations.

The results presented here show that the both reported heuristic approaches [4], [5], provide results that are in most cases either yield the same as those reported in the original references or superior as seen in problems #3 and #4. However, the optimum result provided by MS solver in most cases agree with the reported one, but it is worse for the test problem # 4, and the search for global optimum for complex

problems requires different initial starting vector as seen for test problem #11. The execution times in most cases are comparable between those for heuristic methods and MS Excel solver except for test problems #2 shows marked improvement of several order of magnitude for the heuristic method [4] based on 3^n enumeration.

TABLE I
COMPARISON OF THE RESULTS FROM HEURISTIC SEARCH METHODS [4], [5] AND MS EXCEL SOLVER

Integer Optimization Problem	Optimum results using heuristic method [4]	Optimum Reported in Reference Paper	Optimum results using heuristic method [5]	Optimum results using Ms Excel solver	CPU time (sec) 3n scheme /tunneling /MS solver
1) Dynamic variable problem [14]	(0,1,2) f= 16	(0,1,2) f= 16	(0,1,2) f= 16	(0,1,2) f = 16	0.01/0.055/0.14
2) Rel'y allocation problem [15]	(1,2,1,1,1,5,5,5,6,6) f=0.9966	(1,2,1,1,1,5,5,5,6,6) f=0.9966	(2,1,1,1,1,5,6,4,6,6) f=0.9943	(1,2,1,1,1,5,5,5,6,6) f=0.9966	0.605/0.109 /899.89
3) Rel'y opt with multiple choice of comp [16]	(2,0,0,1,10,1,0) f=0.975982	(2,0,0,0,3,0,1,0) f=0.97206	(2,0,0,1,10,1,0) f=0.975982	(2,0,0,1,10,1,0) f=0.975982	0.055/0.110/0.437
4) Knapsack Problem [17]	(32,2,1,0,0,0,0) f=19979	(33,1,0,1,0,0,0) f=19972	(32,2,1,0,0,0,0) f=19979	(33,1,0,1,0,0,0) f=19972	0.00/0.769/ 0.062
5) Rosen-Suzuki Problem [18]	(0,1,2,-1) f=-44.0	(0,1,2,-1) f=-44.0	(0,1,2,-1) f=-44.0	(0,1,2,-1) f=-44.0	0.000/0.165/0.374
6) Integer linear problem [19]	(4,87,34,149,0) f= 316	(4,87,34,149,0) f= 316	(4,87,34,148,0) f= 315	(4,87,34,149,0) f= 316	0.110/0.110/0.265
7) Transport Problem [20]	(849,351,0,0,250,0,0,750,1,49,750,0) f= 84020	(850,350,0,0,240,2,0,750,2,40,750,0) f= 84000	(850,350,0,0,248,2,0,750, 2,48,750,0) f= 84056	(850,350,0,0,240,2,0,750 ,2,40,750,0) f= 84000	3.74/55.75/ 0.032
8) Quadratic problem [21]	Could not locate solution	(1,1,1,1,1,1,1,1,1,3,3,3,1) f=15	(1,1,1,1,1,1,1,1,1,3,3,3,1) f=15	(1,1,1,1,1,1,1,1,1,3,3,3,1) f=15	NA/20.16/ 0.047
9) Quadratic constrained problem [21]	(5,1,5,0,5,10) f= -310	(5,1,5.,0,5,10) f= -310	(5,1,5.,0,5,10) f= -310	(5,1,5.,0,5,10) f= -310	000/0.440/ 0.031
10) Linear Integer Problem [22]	(1,1,24,52,0) f=-76	(1,1,24,52,0) f=-76	(1,1,23,53,0) f=-76	(1,1,24,52,0) f=-76	0.016/9.219/ 0.016
10) Linear Integer Problem [22]	(1,1,24,52,0) f=-76	(1,1,24,52,0) f=-76	(1,1,23,53,0) f=-76	(1,1,24,52,0) f=-76	0.016/9.219/0.016
11)Non-linear Integer Problem [22]	(5,4,1,1,6,3,2,0) f=110	(5,4,1,1,6,3,2,0) f=110	(4,4,1,2,6,2,2,0) f=128.00	(5,4,1,1,6,3,2,0) f=110	0.016/0.297/ 5.109
12) Non-linear Integer Problem [23]	(16,22,5,5,7) f=807	(16,22,5,5,7) f=807	(16,22,5,5,7) f=807	(16,22,5,5,7) f=807	0.000/0.110/ 0.172
13) Large Variable Integer Problem [24]	Could not locate solution	(48,73, 16,86 ,49, 99,94,79 ,98, 86,94,33, 95, 80,53, 86, 87, 50,39,78, 47, 72, 97,98, 73, 86, 99, 81, 77, 95,28, 95, 58, 23,55, 70, 35, 82,32,94) f= 1030361	(99,99,….,99) f= 1,352,439	99,99,….,99) f= 1,352,439	NA/0.078/ 0.062

The work reported here clearly demonstrates that the algorithm for heuristic methods [4], [5] to find optimum for general integer programming problems is efficient as demonstrated by the comparison of results with those reported in the reference papers and those provided by the commercially available tool MS Excel solver, and would converge to a global or near global optimum solution.

REFERENCES

[1] H. P. Williams, (2009). Logic and integer programming. International Series in Operations Research & Management Science. **130**. ISBN 978-0-387-92280-5.

[2] D. J. Wilde and C. S. Beightler, Foundations of Optimization, Prentice-Hall, Inc., Englewood Cliffs, New Jersey (1967).

[3] Leon **S.** Lasdon, Richard L. Fox, Margery W. Ratner, Nonlinear Optimization Using the Generalized Reduced Gradient Method, Office of Naval Research: Technical Memorandum No. 325, October 1973, distributed by National Technical Information Service, U. S**.** Department of Commerce.

[4] V. K. Srivastava and A. Fahim, A Two-Phase Optimization Procedure for Integer Programming Problems, An International Journal Computers & Mathematics with Applications, 42, (2001) 1585-1595.

[5] Vijaya K. Srivastava and Davide Spinello. A Two-Phase Combined Gradient-Tunneling Based Algorithm for Constrained Integer Programming Problems, Journal of Information & Optimization Sciences, Vol. 36 (2015), No.4, pp. 339-365: DOI:10.1080/02522667.2014.932092.

[6] A. P. French, A. C. Robinson & J. M. Wilson, Using a Hybrid Genetic-Algorithm/Branch and Bound Approach to Solve Feasibility and Optimization Integer Programming Problems, Journal of Heuristics (2001) 7: 551. doi:10.1023/A:1011921025322.

[7] J. Kennedy, R. Eberhart: Particle swarm optimization (C)// Proceedings of IEEE International Conference on Neural Networks. Perth, WA: IEEE Press, 1995: 1942−1948.

[8] Y. Tan, G. Tan, G. & S. Deng, Hybrid particle swarm optimization with chaotic search for solving integer and mixed integer programming problems J. Cent. South Univ. (2014) 21: 2731. doi:10.1007/s11771-014-2235-6.

[9] A. V. Levy and A. Montalvo, The tunneling algorithm for the global minimization of functions, SIAM J, sci.stat. comput, Vol 6, No 1, 15-29 (January 1985).

[10] A. H. Land and A. Doig, An automatic method of solving discrete programming problems, Econometrica 28, 497-520, (1960).

[11] R. E. Gomory, An all-integer programming algorithm, In Industrial Scheduling, (Edited by J. F. Muth and G. L. Thompson), Chapter 13, Prentice Hall, Englewood Cliffs, N. J. (1963).

[12] S. M. Roberts and H. L. Lyvers, The gradient method in process control, Industrial and Engineering Chemistry 53 (1), 877-882 (November 1961).

[13] W. R. Klingman and D. M. Himmelblau, Nonlinear programming with the aid of a multiple-gradient summation technique, Journal of the Association of Computing Machinery 11, (4),400-415, (October 1964).

[14] G. R. Walsh, Methods of Optimization, John Wiley and Sons, 1975

[15] V.R. Prasad and W. Kuo, Reliability optimization of coherent systems, IEEE Transactions on Reliability, Vol 49, No 3, 323-330 (September 2000).

[16] M. Chern and J. Jon, Reliability optimization problems with multiple constraints, IEEE Transactions on Reliability, R-35 (4), 431-436

(October1986).
[17] G. L. Nemhauser and L. A. Wolsey, Integer programming and combinatorial optimization, p443, John Wiley & Sons, (1988).
[18] F.J. Gould. Nonlinear tolerance programming. In F.A. Lootsma, editor, Numerical Methods for Nonlinear Optimization, pages 349-366 Academic Press, 1988.
[19] H.-N. Huynh, H. J. Connell, and S. Kumar. Random search method for integer programming, Article; article/report, 1987. Online: http://trove.nla.gov.au/work/13720937.
[20] Integer Optimization and the Network Models: http://home.ubalt.edu/ntsbarsh/opre640a/partIII.htm.
[21] C. A. Floudas and P. M. Pardalos. A Collection of Test Problems for Constrained Global Optimization Algorithm, chapter 2. Lecture Notes in Computer Science Springer, 1990.
[22] Q. Zheng, D. Zhuang, Integral global optimization: Algorithms, implementations and numerical tests, Journal of Global Optimization 7 (4) (1995) 421–454.
[23] W. Zhu, H. Fan, A discrete dynamic convexized method for nonlinear integer programming, Journal of Computational and Applied Mathematics 223 (2009) 356–373.
[24] C. Mohan, H. T. Nguyen, A controlled random search technique incorporating the simulated annealing concept for solving integer and mixed integer global optimization problems, Computational Optimization and Applications 14 (1999) 103–132.