รายงานวิจัยฉบับสมบูรณ์

โครงการ การวิจัยขั้นตอนวิธีการหาค่าเหมาะที่สุด

โดย นางสาว ธีรนุช บุนนาค และคณะ

กรกฎาคม 2550

สัญญาเลขที่ MRG4880151

รายงานวิจัยฉบับสมบูรณ์

โครงการ การวิจัยขั้นตอนวิธีการหาค่าเหมาะที่สุด

คณะผู้วิจัย	สังกัด
ธีรนุช บุนนาค	ภาควิชาคณิตศาสตร์ คณะวิทยาศาสตร์ มหาวิทยาลัยเชียงใหม่
Min Sun	Department of Mathematics, University of Alabama
ปิยะพงศ์ เนียมทรัพย์	ภาควิชาคณิตศาสตร์ คณะวิทยาศาสตร์ มหาวิทยาลัยเชียงใหม่

สนับสนุนโดยสำนักงานกองทุนสนับสนุนการวิจัย

Continuous tabu search

Dhiranuch Bunnag[a*], *Min Sun*[b]

[a] *Department of Mathematics, Chiang Mai University, Chiang Mai, Thailand 50200*

[b] *Department of Mathematics, University of Alabama, Tuscaloosa, AL, USA, 35487*

We present a stochastic global optimization algorithm, referred to as a tabu search, for solving unconstrained optimization problems over a compact search domain. It is a real-coded that converges in probability to the optimal solution. We have experimented with several ways in defining “neighborhood.” The theoretical and experimental results show that the tabu search performs better than pure random search.

Keywords— continuous tabu search, convergence in probability

* Corresponding author. Tel.: 66-53-943327 ; fax: 66-53-892280;

E-mail: dhiranuch@yahoo.com

ขั้นตอนวิธีการค้นหาทาบูสำหรับตัวแปรที่เป็นจำนวนจริง

ธีรนุช บุนนาค[a], มิน ซัน[b]*

[a]ภาควิชาคณิตศาสตร์ คณะวิทยาศาสตร์ มหาวิทยาลัยเชียงใหม่50200

[b] ภาควิชาคณิตศาสตร์ มหาวิทยาลัยอะลาบามา รัฐอะลาบามา สหรัฐอเมริกา 35487

เรานำเสนอขั้นตอนวิธีสโทแคสติกสำหรับวิธีการหาค่าเหมาะที่สุดที่เรียกว่า วิธีการค้นหาทาบู เพื่อแก้ปัญหาวิธีการหาค่าเหมาะสมที่ไม่มีเงื่อนไขบังคับ ในโดเมนที่เป็นเซตกระชับ ขั้นตอนวิธีนี้ลู่เข้าด้วยความน่าจะเป็นหนึ่งและใช้กับโดเมนที่เป็นจำนวนจริง เราได้ทดสอบประสิทธิภาพโดยใช้การนิยาม “neighborhood.” แบบต่างๆ ผลทางทฤษฎีและทางการทดลองเชิงตัวเลขแสดงให้เห็นว่าขั้นตอนวิธีนี้ดีกว่าการค้นหาแบบสุ่ม

คำหลัก: การค้นหาทาบู การลู่เข้าในทางความน่าจะเป็น

* ผู้วิจัยสำหรับการติดต่อ โทรศัพท์: 66-53-943327 ; แฟกซ์: 66-53-892280;
อีเมล์: dhiranuch@yahoo.com

หน้าสรุปโครงการ (Executive Summary)

ทุนพัฒนาศักยภาพในการทำงานวิจัยของอาจารย์รุ่นใหม่

---

1. ชื่อโครงการ (ภาษาไทย) การวิจัยขั้นตอนวิธีการหาค่าเหมาะที่สุด

(ภาษาอังกฤษ) Research on global optimization algorithms

2. คณะนักวิจัย

2.1 ชื่อ นามสกุล ธีรนุช บุนนาค (Dhiranuch Bunnag)

คุณวุฒิ PhD

สถานที่ทำงาน ภาควิชาคณิตศาสตร์ คณะวิทยาศาสตร์ มหาวิทยาลัยเชียงใหม่ อ. เมือง จ. เชียงใหม่ 50200

โทรศัพท์ (053)943327 โทรสาร (053)892280 email: dhiranuch@yahoo.com

หน้าที่หรือความรับผิดชอบในโครงการ หัวหน้าโครงการวิจัย

2.2 ชื่อ นามสกุล Min Sun

คุณวุฒิ PhD

สถานที่ทำงาน Department of Mathematics, University of Alabama, Box 870350,Tuscaloosa, AL, 35487-0350

โทรศัพท์ (205) 348-1986 โทรสาร (205) 348-7067 email: msun@gp.as.ua.edu

หน้าที่หรือความรับผิดชอบในโครงการ นักวิจัยที่ปรึกษา

2.3 ชื่อ นามสกุล ปิยะพงศ์ เนียมทรัพย์

คุณวุฒิ PhD

สถานที่ทำงาน ภาควิชาคณิตศาสตร์ คณะวิทยาศาสตร์ มหาวิทยาลัยเชียงใหม่ อ.เมือง จ. เชียงใหม่ 50200

โทรศัพท์ (053) 943327 โทรสาร.(053)892280 email: scipnmsp@chiangmai.ac.th

หน้าที่หรือความรับผิดชอบในโครงการ นักวิจัยที่ปรึกษา

3. สาขาวิชาที่ทำการวิจัย optimization

4. งบประมาณทั้งโครงการ 335,600 บาท

5. ระยะเวลาดำเนินงาน 2 ปี (1 มิถุนายน 2548 - 31 พฤษภาคม 2550)

6. ปัญหาที่ทำการวิจัย และความสำคัญของปัญหา

Global optimization is a task of finding at least one best solution that optimizes a given objective function $\underset{x\in\Omega}{Min}\ f(x)$. Many new theorems, algorithms and computational aspects in global optimization have been used to solve many problems in science and engineering. The applications are for examples finance, allocation and location problems, operations research, statistics, structural optimization, engineering design, network and

transportation problems, chip design and database problems, nuclear and mechanical design, chemical engineering design and control, and molecular biology [1]. Algorithms for solving global optimization problems can be categorized into two classes: the stochastic methods that find the global minimum with high probability and the deterministic methods that guarantee to find a global minimum with desired accuracy. Examples of those methods are the following [3]:

1. Stochastic methods
    - random search
    - clustering methods
    - simulated annealing
    - tabu search
    - genetic algorithms
2. Deterministic methods
    - branch and bound
    - interval methods

We propose our version of continuous tabu search for unconstrained optimization problem $\underset{x\in\Omega}{Min}\ f(x)$ with the convergence in probability. We will experiment with different type of "neighborhood'' to see the performance of the algorithms. The convergence proof of each algorithm will be provided. We might improve the efficiency of our algorithm by using tabu search with other optimization methods. We are also interested in a repair operator for quadratic constrained. A repair algorithm is designed to be an assistant of unconstrained optimization algorithms used for solving constrained problems. The repair problem itself is a distance minimization problem. i.e. if we let $\tilde{x}$ be any given point, we want to find a point *x* nearest to $\tilde{x}$ that satisfy the constraint $x^T Qx \le b$. In other words, we want to solve the problem $\underset{x\in\Omega}{Min}\ \|x-\tilde{x}\|^2$ s.t. $x^T Qx \le b$.

References:

[1] Panos M. Pardalos, H. Edwin Romeijn, and Hoang tuy. Recent developments and trends in global optimization. *Journal of Computational and Applied Mathematics*, 124: 209-228, 2000.

[2] Dhiranuch Bunnag and Min Sun. Genetic Algorithm for Constrained Global Optimization in Continuous Variables, accepted to appear in Applied Mathematics and Computation,

2005.
[3] Aimo Torn and Antanas Zilinskas. *Global Optimization. Lecture Notes in Computer Science, No. 350*. Springer Verlag, Heidelberg, 1989.

7. วัตถุประสงค์

เพื่อพัฒนาขั้นตอนวิธีการค้นหาทาบู (tabu search algorithm) พิสูจน์การลู่เข้า (convergence) และเขียนโปรแกรมทดสอบประสิทธิภาพของขั้นตอนวิธี (algorithm) ที่เสนอ

8. ระเบียบวิธีวิจัย

8.1 ทำการรวบรวมและทบทวนเอกสารที่เกี่ยวข้องกับปัญหาที่จะทำการวิจัย

8.2 ศึกษาเอกสารเพิ่มเติมเพื่อหาเทคนิคและผลการวิจัยที่สำคัญและเกี่ยวข้องกับปัญหาที่ต้องการทำวิจัย

8.3 ออกแบบขั้นตอนวิธี แบบต่างๆ

8.4 เขียนโปรแกรม (ใช้ C++) เพื่อทดสอบประสิทธิภาพของขั้นตอนวิธี

8.5 พิสูจน์การลู่เข้าของขั้นตอนวิธี

8.6 เสนอบทความตีพิมพ์ในวารสารระดับนานาชาติ

Keywords : continuous tabu search, convergence in probability

9. ผลงานจากโครงการวิจัยที่ได้รับทุนจาก สกว.

บทความ continuous tabu search กำลังอยู่ในระหว่างการปรับปรุงแก้ไขเพื่อส่งตีพิมพ์ใน Applied mathematics and computation

รายละเอียดการทำวิจัย
บทความที่กำลังอยู่ในระหว่างการปรับปรุงแก้ไขเพื่อส่งตีพิมพ์

Continuous Tabu Search
Dhiranuch Bunnag, Min Sun

1. General idea of tabu search

We consider this optimization problem

$$\text{minimize } f(x) : x \in \Omega, \tag{1}$$

where f is the function to be optimized and Ω is a set of the feasible solutions or the search domain. Tabu search (TS) is an iterative method originally designed for combinatorial optimization problems. TS was introduced by Glover [4] and has been successfully used to solve a wide range of problems such as scheduling, time-tabling, graph coloring, the cutting stock problem, the Knapsack problem, and the Traveling Salesman Problem.

The main idea of TS is to prevent the search from being trapped at locally optimal solutions by prohibition of backward moves. A typical step of tabu search begins by searching for a better candidate in a neighborhood of the current solution. To avoid repeating the steps just used earlier, the method records recent moves in one or more tabu lists. The intent of the list is to prevent the previous moves from being repeated and to ensure that it is not reversed. One drawback of the tabu list is that we may get a tabu list which prevents us from searching intensively in a promising region. If that happens we will relax the tabu status by overruling it. It will be prescribed by *aspiration conditions*.

An outline of a general tabu search procedure may be described as follows.

1. Choose an initial solution x^0. Set $xbest = x^0$ (the currently available best solution) and $k = 0$.
2. Generate a subset V of candidates in a neighborhood of x^k $(N(x^k))$ such that either none of the tabu conditions is violated or at least one of the aspiration conditions is satisfied.
3. Choose the best in V (with respect to $f(x)$), and call it x^{k+1}.
4. If $f(x^{k+1}) < f(xbest)$ then set $xbest = x^{k+1}$.
5. Update the tabu list and aspiration conditions. Set $k = k+1$.
6. If a stopping condition is fulfilled then stop. Otherwise go to step 2.

Stopping conditions could be the following:

- The number of iterations is larger than the maximum number of iterations allowed.
- No more candidates can be generated in a neighborhood, i.e. $V = \varnothing$.
- The number of iterations since the last improvement of $xbest$ has reached a predetermined number.
- The objective function value reaches a pre-specified threshold value.

The choice of V and the definition of neighborhood of x are crucial to the effectiveness of TS. Glover and Hanafi provide an upper bound for the number of iterations for the discrete tabu search in [5]. The idea of the discrete TS has been adapted to the continuous case as reported in Chelouah and Siarry [2] , Battiti and Tecchiolli [1], and Cvijovic and Klinowski [3]. <<briefly summarize them in the framework of the

general TS above

DB:
Cvijovic and Klinowski [3] assume solution space to be hypercube in R^n , n is a number of variables. Partition each axis x_i to be p_i parts. Thus the solution space is divided into cells. In each iteration, the neighborhood is defined to be those sample points from a uniform distribution over a given number of randomly chosen cells. There are two kinds of prohibit moves; if it produces a solution which we have already seen it in the previous L iterations and if it results in a higher objective function value than some specified value. The aspiration condition is the current best value of the objective function found so far.

Chelouah and Siarry [2] call the algorithm Enhanced Continuous Tabu Search (ECTS). They define a hyperrectangular space centered on a point $s = (s_1,s_2,...,s_n)$ to be a set $R(s,r) = \{ x = (x_1,x_2,...,x_n) \mid |x_i - s_i| < r;\ 1 \le i \le n$. Consider a set of hyperrectangles $R(s,r)$ centered on the current solution s with different r; $h_0, h_1, h_2,..., h_v$ where $h_{i-1} < h_i$. The space is partitioned into v subspaces $C_i(s,h_{i-1},h_i) = R(s,h_i) \setminus R(s,h_{i-1}),\ 1 \le i \le v$. The neighbors of s is defined to be the v points randomly selected from inside of each C_i $1 \le i \le v$. The tabu region is the union of all the hyperrectangles centered on those points in tabu list. All newly generated neighbors must not be in the tabu region.

Battiti and Tecchiolli [1] describe the search region as a box. The bound of each x_i is described by $L_i < x_i < U_i$ where i = 1,2,...,N. Divide the initial box to be 2^N equal size boxes. Each box is evaluated the value f(B) by two possible ways; one is using the average of the function values of the sample points from that box and the other is the minimum of the function values among the sample points. The best box will be subdivided into 2^N equal size boxes and the identification of this best box will be kept in tabu list. The neighbors are defined to be the randomly chosen boxes, so those sample boxes can be in different size. The length of the tabu list is adapted during the search. The local minimum is obtained by using the shaker algorithm.

There are other hybrid methods such as in Chelouah and Siarry [9] and Hedar and Fukushima [10] use tabu search with Nelder-Mead simplex algorithm. The tabu search is used to find the promising area and then search this area by using simplex algorithm.

The articles related to the convergence of tabu search are for examples Hanafi [7] studied the convergence of the combinatorial optimization problem using the scheme with two assumptions; *i)* $x \in N(x') \Leftrightarrow x' \in N(x)$ for all x and x' in the search domain E. and ii) for every pair of solutions x and x' in E there exists a path from x to x'. It was shown that if E is finite and the two assumptions hold then the tabu search terminates after exploring all solutions in E.

For the multiple-minima problem of continuous functions Ji and Tang [8] proposed the memory tabu search (MTS) which yields a sequence of solution that converges with probability one to the optimal solution. The MTS define neighborhood to be the whole search domain. The new move is generated by uniform distribution (or Gaussian distribution). The tabu moves are considered by three criteria:
1) the total distance moved at the current iteration.
2) the total change in the objective function.

3) the percentage improvement or destruction.

The following two are added to the references.

[9] Rachid chelouah and Patrick Siarry, A hybrid method combining continuous tabu search and Nelder-Mead simplex algorithms for global optimization of multiminima functions, European Journal of Operational Research 2003

[10] Abdel-Rahman Hedar and Masao Fukushima, Tabu search directed by direct search methods for nonlinear global optimization, European Journal of Operational Research 2004
>>

This article addresses TS in the continuous case. After presenting a more or less standard TS for the continuous optimization, we provide four <<three? DB: It is three. >> additional variations. For all of them, we offer some theoretical convergence analysis. Under mild assumptions, the TS algorithms are shown to converge at least in probability. Near the end, we provide some numerical testing results of those algorithms along with comparison with the pure random search.

2. Our simple continuous tabu search: TS1

A simple continuous tabu search is introduced and analyzed in this section. Assume that the search domain Ω is a hypercube in $R^n, l \le x \le b$, and that there is a unique globally optimal solution. We first introduce variables and parameters used in our algorithm. <<consider the case when #(x*) is finite? Yes in the previous sentence we assume unique globally solution. I think the argument is valid when the number of x* is finite. MS: if so, relax the assumption above and justify accordingly. You may keep the original proof for #(x*)=1 and provide extension arguments.>>

- Partition the ith direction of x into p_i parts, $i = 1,...,n$. Thus the entire search domain is divided into $N_c = p_1 p_2 \cdots p_n$ cells denoted as $\{1,2,...,N_c\}$.
- N_s = number of samples from each cell.
- y^k = a solution that has the minimum value of f among all the candidates discovered by the algorithm after k iterations.
- *fbest* at the k th iteration is $f(y^k)$.
- L_T = tabu list size.
- *tabu_list* = a vector of length L_T that contains the names of tabu cells. We address them by integers in $\{1,2,...,N_c\}$.
- $L_T^{'}$ = number of distinct cells in *tabu_list*; $1 \le L_T^{'} \le L_T$.

Algorithm 1: TS1

1. Set up parameters L_T, N_s, and *List* = all cells.
2. Set iteration counter *k = 0*, and $tabu\_list = \varnothing$.

3. Randomly select a point x^k.

4. Set *fbest* = f(x^k), $y^k = x^k$.

5. **Repeat**

flag =1 (search indicator for the new *x*)

While flag = 1 **do**

Call candidate(List,N_s, *icell*, $\tilde{x}$, $\tilde{f}$).

If (icell $\notin$ *tabu_list*) **or** ($\tilde{f} <$ *fbest*) **then**

Put *icell* in *tabu_list* (might appear more than once).

$x^{k+1} = \tilde{x}$.

flag = 0.

If $\tilde{f} <$ *fbest* **then**

$y^{k+1} = \tilde{x}$ (update the best solution that the algorithm just discovered).

fbest =f($\tilde{x}$).

else $y^{k+1} = y^k$.

Update *k*: *k* = *k*+*1*.

end if

end while

until one of stopping criteria is satisfied.

Subroutine candidate(*List*, N_s, *icell*, $\tilde{x}$, $\tilde{f}$)

Input: List, N_s.

Output: *icell*, $\tilde{x}$, $\tilde{f}$.

1. Randomly select a cell from the input *List* and call it *icell*.
2. Randomly select N_s sample points from this cell.
3. Evaluate *f* of each sample.
4. Find the minimum among the samples (minimum point $\tilde{x}$, minimum value $\tilde{f} = f(\tilde{x})$).
5. Return *icell*, $\tilde{x}$, $\tilde{f}$.

Throughout the article, the randomness is with respect to an underlying probability space. As usual, the capital letter X could be used to denote the random variable representing the random sample, while the lower case x has been used to represent a particular realization of X. In our implementations of TS, we use the *tabu_list* of fixed length L_T. We start out with an empty list. After L_T iterations the list will be full. After that we replace the oldest element in the list with the incoming element that the algorithm has just detected. The same strategy is applied to any other list of a limited length.

Let x^* be the unique globally optimal solution of (1) and

$$B_f(x^*, \varepsilon_f) = \{x \in \Omega : | f(x) - f(x^*) | < \varepsilon_f\}.$$

We also assume <<assume continuity of f? DB: yes >> that $m(B_f(x^*, \varepsilon_f)) > 0$ for any $\varepsilon_f > 0$, where $m(A)$ means the Lebesgue measure of a set A. Two sequences $\{x^k\}$ and $\{y^k\}$ are produced by TS1. The sequence $\{x^k\}$ shows movements of the algorithm.

While a sequence $\{y^k\}$ keeps track of the best x found among all the samples from the first iteration to the current iteration k. Our goal is to show that

$$P(\{Y^k\}_{k=1}^{t} \cap B_f(x^*, \varepsilon_f) = \varnothing) \to 0 \text{ as t} \to \infty.$$

Explanation of the terms that we will use in our proof:

- t = the current iteration counter.
- c^* = a cell which contains x^*.
- $\beta(t)$ = the desired level of the f value at iteration t or the current value of *fbest*.
- $P_{rej}(\beta(t))$ = Probability of rejecting a cell that is contained in *tabu_list*

  <<DB: I need to remove "that is contained in *tabu_list*" in order to use it in the proof of TS3. >>

  at level $\beta(t)$. Suppose x is a minimum among the random samples from the cell. If $f(x) \geq \beta(t)$, reject this cell with probability $P_{rej}(\beta(t))$. Therefore, $P_{rej}(f(x^*)) = 1$ and $P_{rej}(\infty) = 0$.
- $P(c^* \notin tabu\_list) = P(c^*$ has never been chosen in the past L_T iterations)
  $= (\frac{N_c - 1}{N_c})^{L_T}$.
- Probability of selecting a cell and obtaining c^* is $\frac{1}{N_c}$.
- Probability of obtaining a point in $B_f(x^*, \varepsilon_f)$ when selecting a sample point in c^* is $\frac{m(B_f(x^*, \varepsilon_f) \cap c^*)}{m(c^*)}$.
- In each iteration we sample N_s points in the candidate *icell* independently. Supposing $icell = c^*$, the probability that at least one of the points from c^* is contained in $B_f(x^*, \varepsilon_f)$ is $\frac{1}{N_c}(1 - (1 - \frac{m(B_f(x^*, \varepsilon_f) \cap c^*)}{m(c^*)})^{N_s})$.
- Probability of selecting a cell and obtaining a cell from *tabu_list* is $\frac{L_T'}{N_c} \geq \frac{1}{N_c}$.

We would like to find or estimate the probability that TS1 yields $x^t \in B_f(x^*, \varepsilon_f)$. First consider the following dichotomy.

I: $B_f(x^*, \varepsilon_f) \subseteq c^*$. For each iteration counter *t*, one of the following events must have occurred in order to have $y^t \in B_f(x^*, \varepsilon_f)$.

- Select a cell and obtain c^*. Sample N_s points and at least one of the points is contained in $B_f(x^*, \varepsilon_f)$. This event occurs with probability $\frac{1}{N_c}(1 - (1 - \frac{m(B_f(x^*, \varepsilon_f))}{m(c^*)})^{N_s})$.
- Select a cell and obtain a cell which is contained in *tabu_list*. Reject the cell with probability $P_{rej}(\beta(t))$. Sample a new cell and obtain c^*. Sample N_s points from the cell c^*, with at least one of the samples contained in $B_f(x^*, \varepsilon_f)$. The probability for this event is
  $$\frac{L_T'}{N_c} \cdot P_{rej}(\beta(t)) \cdot \frac{1}{N_c}(1 - (1 - \frac{m(B_f(x^*, \varepsilon_f))}{m(c^*)})^{N_s}).$$
- The first two selected cells are contained in *tabu_list*. Both cells are rejected individually with probability $P_{rej}(\beta(t))$. However in the selection of the third cell

we get c^*. Among our N_s samples, at least one of the samples is contained in $B_f(x^*, \varepsilon_f)$. The probability for this event is at least

$$(\tfrac{1}{N_c} \cdot P_{rej}(\beta(t)))^2 \cdot \tfrac{1}{N_c}(1-(1-\tfrac{m(B_f(x^*,\varepsilon_f))}{m(c^*)})^{N_s}).$$

- The process might continue with more rejections of cells from *tabu_list* before obtaining c^*. The probability of rejecting k $(k \geq 3)$ cells before obtaining a cell that has at least one sample belonging to $B_f(x^*, \varepsilon_f)$ is at least

$$(\tfrac{1}{N_c} \cdot P_{rej}(\beta(t)))^k \cdot \tfrac{1}{N_c}(1-(1-\tfrac{m(B_f(x^*,\varepsilon_f))}{m(c^*)})^{N_s}).$$

Therefore, we can conclude that

$$\begin{aligned} P(X^t \in B_f(x^*,\varepsilon_f)) &\geq \tfrac{1}{N_c}(1-(1-\tfrac{m(B_f(x^*,\varepsilon_f))}{m(c^*)})^{N_s}) \\ &+ \tfrac{1}{N_c} \cdot P_{rej}(\beta(t)) \cdot \tfrac{1}{N_c} \cdot (1-(1-\tfrac{m(B_f(x^*,\varepsilon_f))}{m(c^*)})^{N_s}) \\ &+ (\tfrac{1}{N_c} \cdot P_{rej}(\beta(t)))^2 \cdot \tfrac{1}{N_c} \cdot (1-(1-\tfrac{m(B_f(x^*,\varepsilon_f))}{m(c^*)})^{N_s}) + ... \\ &\geq \frac{1}{N_c}(1-(1-\frac{m(B_f(x^*,\varepsilon_f))}{m(c^*)})^{N_s})[1+\sum_{k=1}^{\infty}(\frac{1}{N_c} \cdot P_{rej}(\beta(t)))^k] \quad (2) \\ &\geq \frac{1}{N_c}(1-(1-\frac{m(B_f(x^*,\varepsilon_f))}{m(c^*)})^{N_s}) > 0. \quad (3) \end{aligned}$$

II: $B_f(x^*, \varepsilon_f) \subseteq c_1^* \cup c_2^* \cup \cdots c_q^*$ where $m(c_j^* \cap B_f(x^*, \varepsilon_f)) \neq 0$ for $j = 1,...,q$ with some $q \geq 2$. In other words, $B_f(x^*, \varepsilon_f)$ is not entirely contained in a single cell. Let us introduce a new set of notations:

- $B_j = c_j^* \cap B_f(x^*, \varepsilon_f)$. Thus $B_f(x^*, \varepsilon_f) = \cup_{j=1}^{q} B_j$.
- Let $\frac{m(B_k)}{m(c_k^*)} = \min \frac{m(B_j)}{m(c_j^*)} : j = 1,...,q$ <<min{…: j=1,…,q}>>, so that $\frac{m(B_k)}{m(c_k^*)} \leq \frac{m(B_j)}{m(c_j^*)}$ for $j = 1,...,q$.

In view of the arguments used in the former case, we examine the following possible situations.

- Select a cell and obtain c_j^* for some j in {1,…,q}. Then sample N_s points from c_j^* and at least one of the samples is contained in $B_f(x^*, \varepsilon_f)$. This event can occur with probability of at least $\frac{q}{N_c}(1-(1-\frac{m(B_k)}{m(c_k^*)})^{N_s})$.
- Select a cell and obtain a cell which is contained in *tabu_list*. Reject this cell with probability $P_{rej}(\beta(t))$ and then sample a new cell. This time obtain one of c_j^* and at least one of the samples is contained in $B_f(x^*, \varepsilon_f)$. The probability for this event is at least $\frac{1}{N_c} \cdot P_{rej}(\beta(t)) \cdot \frac{q}{N_c}(1-(1-\frac{m(B_k)}{m(c_k^*)})^{N_s})$.
- The first two selected cells are contained in *tabu_list*. Both cells are rejected. For the third selection we get one of the c_j^* and at least one of the samples from this cell is contained in $B_f(x^*, \varepsilon_f)$. The probability for this event is

at least $(\frac{1}{N_c}\cdot P_{rej}(\beta(t)))^2\cdot\frac{q}{N_c}(1-(1-\frac{m(B_k)}{m(c_k^*)})^{N_s})$.

The situation might continue in this manner with more rejections of cells from *tabu_list* before obtaining some c_j^*. Therefore, we can conclude that

$$\begin{aligned}P(X^t\in B_f(x^*,\varepsilon_f))&\geq\tfrac{q}{N_c}(1-(1-\tfrac{m(B_k)}{m(c_k^*)})^{N_s})\\&\quad+\tfrac{1}{N_c}\cdot P_{rej}(\beta(t))\cdot\tfrac{q}{N_c}\cdot(1-(1-\tfrac{m(B_k)}{m(c_k^*)})^{N_s})\\&\quad+(\tfrac{1}{N_c}\cdot P_{rej}(\beta(t)))^2\cdot\tfrac{q}{N_c}\cdot(1-(1-\tfrac{m(B_k)}{m(c_k^*)})^{N_s})+....\\&\geq\frac{q}{N_c}(1-(1-\frac{m(B_k)}{m(c_k^*)})^{N_s})[1+\sum_{i=1}^{\infty}(\frac{1}{N_c}\cdot P_{rej}(\beta(t)))^i]\end{aligned}\qquad(4)$$

The summation $\sum_{i=1}^{\infty}(\frac{1}{N_c}\cdot P_{rej}(\beta(t)))^i$ <<delete]>> converges. Therefore,

$$P(X^t\in B_f(x^*,\varepsilon_f))\ \geq\frac{q}{N_c}(1-(1-\frac{m(B_k)}{m(c_k^*)})^{N_s})>0.$$

(5)

From the dichotomy explained above, we can find a lower bound of the probability that any X^t belongs to $B_f(x^*,\varepsilon_f)$, $P(X^t\in B_f(x^*,\varepsilon_f))$, as follows:

$$P(X^t\in B_f(x^*,\varepsilon_f))\geq\begin{cases}\dfrac{1}{N_c}(1-(1-\dfrac{m(B_f(x^*,\varepsilon_f))}{m(c^*)})^{N_s}) & \text{if}\quad B_f(x^*,\varepsilon_f)\subseteq c^*\\ \dfrac{q}{N_c}(1-(1-\dfrac{m(B_k)}{m(c_k^*)})^{N_s}) & \text{if}\quad B_f(x^*,\varepsilon_f)\subseteq\cup_{j=1}^{q}c_j^*.\end{cases}$$

Define P_B to be the lower bound of $P(X^t\in B_f(x^*,\varepsilon_f))$

$$P_B=\min\{\frac{1}{N_c}(1-(1-\frac{m(B_f(x^*,\varepsilon_f))}{m(c^*)})^{N_s}),\frac{q}{N_c}(1-(1-\frac{m(B_k)}{m(c_k^*)})^{N_s})\}.\qquad(6)$$

From the algorithm the sequence $\{y^k\}$ has the following properties:

1. $f(y^{t+1})\leq f(y^t)$.
2. If $y^t\in B_f(x^*,\varepsilon_f)$, $y^i\in B_f(x^*,\varepsilon_f)$ for $i=t+1,t+2,....$
3. Suppose $x^t\in B_f(x^*,\varepsilon_f)$ at iteration *t*, we will then have $y^t\in B_f(x^*,\varepsilon_f)$.

We therefore conclude that

$P(Y^i\in B_f(x^*,\varepsilon_f))=P(X^i\in B_f(x^*,\varepsilon_f))$ if $i=1,...,t-1$ and

$P(Y^i\in B_f(x^*,\varepsilon_f))\geq P(X^i\in B_f(x^*,\varepsilon_f))$ if $i\geq t$.

This means once the algorithm has found y^t contained in $B_f(x^*,\varepsilon_f)$, the sequence $\{y^k\}_{k=t+1}^{\infty}$ will stay inside $B_f(x^*,\varepsilon_f)$. This analysis now leads to our main result below.

Theorem 2.1 $\forall\,\varepsilon_f>0, P(\{y^k\}_{k=1}^{t}\cap B_f(x^*,\varepsilon_f)=\varnothing)\to 0$. <<$Y^t$ not changed? DB: as $t\to\infty$ >>

Proof. At each iteration t, $P(Y^t\in B_f(x^*,\varepsilon_f))$ depends on the level of $\beta(t)$ or $f(xbest)$ as shown in the term $P_{rej}(\beta(t))$ of Equations (2) and (4). However, we have found a lower bound (P_B from (6)) of this probability which does not depend on iteration number.

$$P(\{Y^i\}_{i=1}^t \cap B_f(x^*,\varepsilon_f) = \varnothing) = P(\text{None of } Y^i \text{ belongs to } B_f(x^*,\varepsilon_f), \forall\, i \le t)$$
$$= \prod_{i=1}^t [1 - P(Y^i \in B_f(x^*,\varepsilon_f))]$$
$$\le \prod_{i=1}^t [1 - P(X^i \in B_f(x^*,\varepsilon_f))]$$
$$\le [1 - P_B]^t \to 0 \text{ as } t \to \infty .$$

. <<break into 2 separate lines in order to have the point in the right spot>>

This completes the proof of convergence in probability of Algorithm 1. □

Since the size of $B_f(x^*,\varepsilon_f)$ is generally a lot smaller than the size of the partitioned cells, we can assume that $B_f(x^*,\varepsilon_f)$ is entirely contained in one cell. Thus only the first case above is our major concern. We define $\eta = \dfrac{m(B_f(x^*,\varepsilon_f))}{m(\Omega)}$. If pure random search is used, the probability that we will find a point contained in $B_f(x^*,\varepsilon_f)$ is η, that is $P(X^t \in B_f(x^*,\varepsilon_f)) = \eta$.

Proposition 2.1 Let $\{x^t\}$ and $\{y^t\}$ be generated by Algorithm 1. If $B_f(x^*,\varepsilon_f)$ is entirely contained in one cell, then

$$P(Y^t \in B_f(x^*,\varepsilon_f)) \ge P(X^t \in B_f(x^*,\varepsilon_f)) \ge \tilde{\square} \qquad (7)$$

Proof. According to (3), it suffices to show

$$\frac{1}{N_c}\left(1 - \left(1 - \frac{m(B_f(x^*,\varepsilon_f))}{m(c^*)}\right)^{N_s}\right) \ge \eta .$$

We will use induction on N_s.

$$N_s = 1;\ \frac{1}{N_c}\left(1 - \left(1 - \frac{m(B_f(x^*,\varepsilon_f))}{m(c^*)}\right)\right) = \frac{m(B_f(x^*,\varepsilon_f))}{N_c m(c^*)} = \frac{m(B_f(x^*,\varepsilon_f))}{m(\Omega)} = \eta$$

$$N_s = 2;$$

$$\frac{1}{N_c}\left(1 - \left(1 - \frac{m(B_f(x^*,\varepsilon_f))}{m(c^*)}\right)^2\right) = 2\frac{m(B_f(x^*,\varepsilon_f))}{N_c m(c^*)} - \frac{m(B_f(x^*,\varepsilon_f))m(B_f(x^*,\varepsilon_f))}{N_c m(c^*)m(c^*)}$$
$$= 2\eta - \eta\frac{m(B_f(x^*,\varepsilon_f))}{m(c^*)}$$
$$= \eta\left(2 - \frac{m(B_f(x^*,\varepsilon_f))}{m(c^*)}\right) \ge \eta$$

Suppose $\dfrac{1}{N_c}\left(1 - \left(1 - \dfrac{m(B_f(x^*,\varepsilon_f))}{m(c^*)}\right)^{N_s}\right) \ge \eta$

Consider

$$\frac{1}{N_c}\left(1 - \left(1 - \frac{m(B_f(x^*,\varepsilon_f))}{m(c^*)}\right)^{N_s+1}\right)$$
$$= \frac{1}{N_c}\left(1 - \left(1 - \frac{m(B_f(x^*,\varepsilon_f))}{m(c^*)}\right)^{N_s}\left(1 - \frac{m(B_f(x^*,\varepsilon_f))}{m(c^*)}\right)\right)$$
$$= \frac{1}{N_c}\left(1 - \left(1 - \frac{m(B_f(x^*,\varepsilon_f))}{m(c^*)}\right)^{N_s} + \frac{m(B_f(x^*,\varepsilon_f))}{m(c^*)}\left(1 - \frac{m(B_f(x^*,\varepsilon_f))}{m(c^*)}\right)^{N_s}\right)$$

$$= \frac{1}{N_c}(1-(1-\frac{m(B_f(x^*,\varepsilon_f))}{m(c^*)})^{N_s}) + \frac{m(B_f(x^*,\varepsilon_f))}{N_c m(c^*)}(1-\frac{m(B_f(x^*,\varepsilon_f))}{m(c^*)})^{N_s}$$

$$= \eta + \eta(1-\frac{m(B_f(x^*,\varepsilon_f))}{m(c^*)})^{N_s}$$

$$= \eta(1+(1-\frac{m(B_f(x^*,\varepsilon_f))}{m(c^*)})^{N_s}) \geq \eta$$

This completes the proof. □

<<Still true if $B_f(x^*,\varepsilon_f)$ is NOT entirely contained in one cell?

DB: I cannot show that it is true when B_f does not contained entirely in one cell. It seems that the following held.

1. If the number of N_s increases, the probability is higher.
2. When using an approximation in (5), it seems to suggest that the process of selecting cells before getting sample points reduce the probability of getting the points from B_f, since the selected cell may contain only a portion of B_f.

>>

TS1 might behave much like a pure random search although it is shown not to be worse. In practice it could take quite some times before y^t gets closer to the global solution x*. The algorithm lacks an intensification procedure that a successful global optimization should have. We have tried to improve the algorithm in three different ways without much change in the proof. The results will be shown in the next three sections after the explanation of each modification.

3. A modified continuous tabu search: TS2

In order to improve and speed up TS1, we expand the neighborhood by allowing more sampling of cells in subroutine candidate2. We will randomly select *ncell* cells instead of only one cell and allow the movement in the direction of the lowest f among all the samples. To refine the solution, we keep a list of length L_p of the best up-to-date y^t. This list is called *promising_list*. A search is performed by sampling some points in a neighborhood of y^t from the *promising_list* when the original algorithm stops improving. This part will not effect the convergence of the algorithm. The algorithm is stated as follows.

Algorithm 2: TS2

1. Set up parameters $L_T, L_p, N_s, ncell$, $List$ = all cells.
2. Set iteration counter k = 0, $promising\_list = \varnothing$, and $tabu\_list = \varnothing$.
3. Randomly select a point x^k.
4. Set fbest = f(x^k), $y^k = x^k$.
5. **Repeat**

   flag =1 (search indicator for the new x).

   While flag = 1 **do**

   Call candidate2 (List,$N_s, ncell, icell, \tilde{x}, \tilde{f}$).

   If (icell $\notin tabu\_list$) **or** ($\tilde{f} < fbest$) **then**

   Put *icell* in *tabu_list* (might appear more than once).

x$^{k+1} = \tilde{x}$.
flag = 0.
If $\tilde{f} <$ *fbest* **then**
$y^{k+1} = \tilde{x}$ (update the best solution that the algorithm just discovered).
fbest = $\tilde{f}$.
If y$^{k+1} \notin$ *promising_list* **then** put y^{k+1} in *promising_list* .
else y$^{k+1} = y^k$.
Update $k: k = k+1$.
end if
end while
until one of stopping criteria is satisfied.

6. Search for a better solution (if any) from the neighborhood of each point contained in *promising_list*.

Subroutine candidate2($List, N_s, ncell, icell, \tilde{x}, \tilde{f}$)

Input: List, N$_s$, *ncell* .

Output: $icell, \tilde{x}, \tilde{f}$.

1. Randomly select *ncell* different cells from the input *List* .
2. In each cell randomly select N$_s$ samples.
3. Evaluate f of all samples.
4. Find the minimum among the samples (minimum point $\tilde{x}$, minimum value $\tilde{f} = f(\tilde{x})$). and use the corresponding cell of $\tilde{x}$ as *icell* .
5. Return $icell, \tilde{x}, \tilde{f}$.

Its analysis of convergence again starts with considering two cases.

Case I: $B_f(x^*, \varepsilon_f) \subseteq c^*$. In each iteration, one of the following events will occur.

- *icell* $= c^*$ and when sampling N_s points from *icell* we found that at least one of the samples is contained in $B_f(x^*, \varepsilon_f)$. This event occurs with probability $\frac{ncell}{N_c}(1-(1-\frac{m(B_f(x^*,\varepsilon_f))}{m(c^*)})^{N_s})$.
- The first *icell* is contained in *tabu_list* and this *icell* is rejected. The probability that this candidate *icell* is rejected is the same <<?

DB: This is wrong. I want to say that if this *icell* is rejected, then all the *ncell* sample cells will be automatically rejected too. This is because of the routine candidate2. Even though some of them may not contained in *tabu_list*. In this paragraph there is nothing to do with the probability yet, the probability will be calculated below this.

>> as that for any other of *ncell* sample cells to be rejected. Then the algorithm repeats the function call of candidate2. A second candidate *icell* is returned and this time *icell* $= c^*$. At least one of the samples from this *icell* is contained in $B_f(x^*, \varepsilon_f)$. Note that when the first *icell* is rejected, it doesn't mean that the other *ncell* -1 cells are also contained in *tabu_list* . The following three steps show how to calculate the probability for this event.

Step 1. Obtain at least one cell from *tabu_list* when randomly select *ncell* cells. Let

p_1 = P (at least one cell is contained in *tabu_list*)

= $1 - P$ (none is contained in *tabu_list*).

Thus

$$p1 = 1 - \frac{\binom{N_c - L_T'}{ncell}}{\binom{N_c}{ncell}}. \text{ <<subscript>>}$$

Step 2. Reject the first *icell* returned from candidate2 since it is contained in *tabu_list*. Let *nl* be the number of cells (from *ncell* cells) which is contained in *tabu_list* ($1 \le nl \le ncell$) and C be the set of those *nl* cells. Let f^b denote the best value of f among the samples from C. Note that f^b depends on *t*.

P (reject the first *icell* returned from candidate2)

$= p_1 \cdot (P_{rej}(f^b))^{ncell-nl} \cdot (P_{rej}(\beta(t)))^{nl}$.

Step 3. Select a new neighborhood by calling candidate2 again and obtain $icell = c^*$. At least one of the samples from this *icell* is contained in $B_f(x^*, \varepsilon_f)$.

Therefore, the probability for this event to occur is

$$p_1 \cdot (P_{rej}(f^b))^{ncell-nl} \cdot (P_{rej}(\beta(t)))^{nl} \cdot \frac{ncell}{N_c} \cdot (1 - (1 - \frac{m(B_f(x^*, \varepsilon_f))}{m(c^*)})^{N_s}).$$

- The process might continue with k ($k \ge 2$) rejections of cells from *tabu_list* before obtaining c^*. Let nl_i be the number of cells (from *ncell* cells) which are contained in *tabu_list* ($1 \le nl_i \le ncell$) at the i th neighborhood selection (by calling candidate2). C_i is a set of those nl_i cells. f_i^b = the best value of f among the samples from C_i. The probability of rejecting k ($k \ge 2$) cells before obtaining a cell that has at least one sample belonging to $B_f(x^*, \varepsilon_f)$ is

$$p_1 \cdot \prod_{i=1}^{k} [(P_{rej}(f_i^b))^{ncell-nl_i} \cdot (P_{rej}(\beta(t)))^{nl_i}] \cdot \frac{ncell}{N_c} \cdot (1 - (1 - \frac{m(B_f(x^*, \varepsilon_f))}{m(c^*)})^{N_s}).$$

Therefore, we can conclude that

$$\begin{aligned}
&P(X^t \in B_f(x^*, \varepsilon_f)) \\
&\ge \frac{ncell}{N_c}(1 - (1 - \frac{m(B_f(x^*, \varepsilon_f))}{m(c^*)})^{N_s}) \\
&\quad + p_1 \cdot (P_{rej}(f^b))^{ncell-nl} \cdot (P_{rej}(\beta(t)))^{nl} \cdot \frac{ncell}{N_c} \cdot (1 - (1 - \frac{m(B_f(x^*, \varepsilon_f))}{m(c^*)})^{N_s}) \\
&+ \ldots + p_1 \cdot \prod_{i=1}^{k} [(P_{rej}(f_i^b))^{ncell-nl_i} \cdot (P_{rej}(\beta(t)))^{nl_i}] \cdot \frac{ncell}{N_c} \cdot (1 - (1 - \frac{m(B_f(x^*, \varepsilon_f))}{m(c^*)})^{N_s}) + \ldots \\
&\quad \ge \frac{ncell}{N_c}(1 - (1 - \frac{m(B_f(x^*, \varepsilon_f))}{m(c^*)})^{N_s}) \\
&\cdot [1 + \sum_{k=1}^{\infty} p_1 \cdot \prod_{i=1}^{k} [(P_{rej}(f_i^b))^{ncell-nl_i} \cdot (P_{rej}(\beta(t)))^{nl_i}]]. \qquad (8)
\end{aligned}$$

Hence

$$P(X^t \in B_f(x^*, \varepsilon_f)) \ge \frac{ncell}{N_c}(1 - (1 - \frac{m(B_f(x^*, \varepsilon_f))}{m(c^*)})^{N_s}) > 0. \qquad (9)$$

Case II: $B_f(x^*,\varepsilon_f) \subseteq c_1^* \cup c_2^* \cup \cdots c_q^*$ where $m(c_j^* \cap B_f(x^*,\varepsilon_f)) \neq 0$ for $j = 1,...,q$ with $q \geq 2$. The following are the possible events in one iteration of the algorithm, leading to the desired event $X^t \in B_f(x^*,\varepsilon_f)$.

- *icell* is one of c_j^* $(j = 1,...,q)$.

  $P($ at least one of the *ncell* sample cells is $c_j^*) = 1-(\frac{N_c-q}{N_c})^q$.

Then we found that at least one of the N_s samples from *icell* is contained in $B_f(x^*,\varepsilon_f)$. This event can occur with probability of at least

$$(1-(\tfrac{N_c-q}{N_c})^q)(1-(1-\tfrac{m(B_k)}{m(c_k^*)})^{N_s}).$$

- The first candidate cell is contained in *tabu_list*. Reject this cell and get a second candidate cell. This time we obtain one of c_j^* $(j = 1,...,q)$ and at least one of the samples from this cell is contained in $B_f(x^*,\varepsilon_f)$. The probability for this event is $p_1 \cdot (P_{rej}(f^b))^{ncell-nl} \cdot (P_{rej}(\beta(t)))^{nl} \cdot (1-(\frac{N_c-q}{N_c})^q)(1-(1-\frac{m(B_k)}{m(c_k^*)})^{N_s})$.
- For the first two times of selecting candidate cells, we obtain two cells which are contained in *tabu_list*. Both cells are rejected. The third time we find one of c_j^* $(j = 1,...,q)$ and at least one of the samples is contained in $B_f(x^*,\varepsilon_f)$. The probability for this event is

  $$p_1 \cdot \prod_{i=1}^{2}[(P_{rej}(f_i^b))^{ncell-nl_i} \cdot (P_{rej}(\beta(t)))^{nl_i}] \cdot (1-(\tfrac{N_c-q}{N_c})^q)(1-(1-\tfrac{m(B_k)}{m(c_k^*)})^{N_s}).$$

The situation might continue in this manner with more rejections of cells from *tabu_list* before obtaining c_j^*. Therefore, we can conclude that

$$\begin{aligned}
&P(X^t \in B_f(x^*,\varepsilon_f)) \\
&\geq (1-(\tfrac{N_c-q}{N_c})^q)(1-(1-\tfrac{m(B_k)}{m(c_k^*)})^{N_s}) \\
&\quad + p_1 \cdot (P_{rej}(f^b))^{ncell-nl} \cdot (P_{rej}(\beta(t)))^{nl} \cdot (1-(\tfrac{N_c-q}{N_c})^q)(1-(1-\tfrac{m(B_k)}{m(c_k^*)})^{N_s}) \\
&\quad + p_1 \cdot \prod_{i=1}^{2}[(P_{rej}(f_i^b))^{ncell-nl_i} \cdot (P_{rej}(\beta(t)))^{nl_i}] \cdot (1-(\tfrac{N_c-q}{N_c})^q)(1-(1-\tfrac{m(B_k)}{m(c_k^*)})^{N_s}) + \ldots \\
&\qquad \geq (1-(\frac{N_c-q}{N_c})^q)(1-(1-\frac{m(B_k)}{m(c_k^*)})^{N_s}) \\
&\qquad \cdot[1+\sum_{k=1}^{\infty} p_1 \cdot \prod_{i=1}^{k}[(P_{rej}(f_i^b))^{ncell-nl_i} \cdot (P_{rej}(\beta(t)))^{nl_i}]]. \qquad (10)
\end{aligned}$$

Consequently,

$$P(X^t \in B_f(x^*,\varepsilon_f)) \geq (1-(\frac{N_c-q}{N_c})^q)(1-(1-\frac{m(B_k)}{m(c_k^*)})^{N_s}) > 0. \qquad (11)$$

From the two cases previously explained, we can find a lower bound of the probability that any x^t belongs to $B_f(x^*,\varepsilon_f)$ as follows:

$$P(X^t \in B_f(x^*,\varepsilon_f)) \geq \begin{cases} \dfrac{ncell}{N_c}(1-(1-\dfrac{m(B_f(x^*,\varepsilon_f))}{m(c^*)})^{N_s}) & \text{if } B_f(x^*,\varepsilon_f) \subseteq c^* \\ (1-(\dfrac{N_c-q}{N_c})^q)(1-(1-\dfrac{m(B_k)}{m(c_k^*)})^{N_s}) & \text{if } B_f(x^*,\varepsilon_f) \subseteq \cup_{j=1}^{q} c_j^*. \end{cases}$$

Define P_B to be the lower bound of $P(X^t \in B_f(x^*,\varepsilon_f))$.

$$P_B = \min\{\frac{ncell}{N_c}(1-(1-\frac{m(B_f(x^*,\varepsilon_f))}{m(c^*)})^{N_s}),(1-(\frac{N_c-q}{N_c})^q)(1-(1-\frac{m(B_k)}{m(c_k^*)})^{N_s})\} \qquad (12)$$

The rest of the proof of convergence in probability is the same as for Algorithm 1.

4. Another modified continuous tabu search: TS3

In the previous two algorithms, TS1 and TS2, we use randomly selected neighborhoods. They are not necessarily near the current cell where the current x is located. We now place some restriction on the movement by defining the neighborhood to be those around the current cell. In other words, the algorithm will move in a more systematic way.

4.1. Neighborhood definition

As mentioned before, we partition each direction i of the original hypercube domain into p_i parts. We address each part as an integer in $\{0,1,..., p_i - 1\}$. Thus each cell will be represented by an integer vector of dimension n. The ith component of a cell vector is the address of the part in direction i. The scheme is shown in Figure 1. A search domain of this problem is $[0,1]\times[0,1]$. The x-direction is partitioned into 5 parts. We label each part as $0,1,2,3,4$. The y-direction is partitioned into 4 parts (labeled as $0,1,2,3$). The cell named *icell* is represented by an integer vector $[2, 2]$. The neighborhood of x^k, where x^k is contained in *icell*, is defined to be all those cells surrounding *icell*, namely [1, 1], [1, 2], [1, 3], [2, 3], [3, 3], [3, 2], [3, 1], [2, 1]. In general, suppose *icell* is represented by $[i_0, i_1, ..., i_n]$. The neighborhood cells are those cells that can be written in the form $[i_0 + \delta_0, i_1 + \delta_1, ..., i_{n-1} + \delta_{n-1}]$ where δ_j takes a value of $-1, 0$, or 1 and $[\square_0, \ldots, \square_{n-1}] \neq 0$. Therefore the total number of all neighborhood cells is $3^n - 1$.

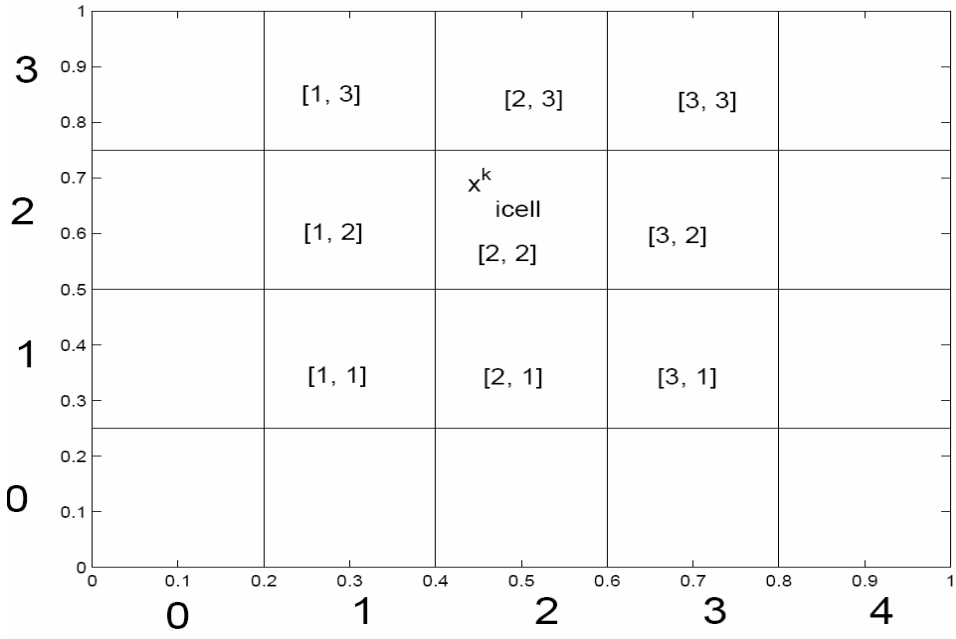

Figure 1. Neighborhood of the current point x^k contained in *icell*

4.2. Statement of Algorithm 3

Algorithm 3: TS3

1. Set up parameters L_T, L_p, N_s.
2. Set iteration counter $k = 0$, $tabu\_list = \varnothing$, and $promising\_list = \varnothing$.
3. Initialization: randomly select a cell (*icell*) and within this cell select a point x^k.
   Set $fbest = f(x^k)$, $y^k = x^k$.
4. **Repeat**
   flag =1 (search indicator for the new x).
   While flag = 1 **do**
5. Select N_s points from each neighborhood cell of the current cell.
   Find the best among the samples (call it $\tilde{x}$).
   Replace *icell* with the address of the cell containing $\tilde{x}$.
6. **If** ($icell \in tabu\_list$) **and** ($\tilde{f} \geq fbest$) **then**
   Randomly select a new *icell* which does not belong to the neighborhood of the current *icell*, *tabu_list*, or *promising_list*. <<Don't sample from this new cell? DB: the above current *icell* should be removed. >>
   end if
7. **If** ($icell \notin tabu\_list$) **or** ($\tilde{f} < fbest$) **then**
   Put *icell* in *tabu_list* (might appear more than once).
   $x^{k+1} = \tilde{x}$.
   flag = 0.
   If $\tilde{f} < fbest$ **then**
   $y^{k+1} = \tilde{x}$ (update the best solution just discovered).
   $fbest = \tilde{f}$.
   If $y^{k+1} \notin promising\_list$ **then** put y^{k+1} in *promising_list*.
   else $y^{k+1} = y^k$.
   Update k: $k = k+1$.
   end if
   end while
   until one of stopping criteria is satisfied.

4.3. Convergence of Algorithm 3

A sequence of $\{y^k\}$ is nonincreasing in f values ($f(y^{k+1}) \leq f(y^k)$). Since the tabu length L_T is relatively short compared to the total number of cells, after a while it will lose its effect on the cells in *tabu_list*. Line 6 in the algorithm allows any cell outside the neighborhood of the current cell, *tabu_list*, and *promising_list* to be selected. With all these features, the probability of missing a positive measure set in the search region is zero. We can apply the convergence of random search from Solis and Wet [6].
<<inconsistency: Why just quote [6] for TS3 while doing direct proof for the other algs? DB: The proof is provided below.

We will first introduce the layer of *icell*. Suppose *icell* is represented by $[i_0, i_1, ..., i_{n-1}]$.
Let C_k be a set of all points contained in the following cells
$[i_0 + \delta_0, i_1 + \delta_1, ..., i_{n-1} + \delta_{n-1}]$ where $\delta_j = 0, \pm 1, ..., \pm$ k and j = 1,2,...,n.
Let L_k be the *k*th layer of the neighborhood of the current cell say *icell*.
$L_1 = (C_1 - icell) \cap \Omega$
$L_k = (C_k - C_{k-1}) \cap \Omega$ where $k = 2,3,...$

The following events will occur.

1. In an initialization stage, we obtain $x^t \in B_f(x^*, \varepsilon_f)$.

$P(X^t \in B_f(x^*, \varepsilon_f))$
= P(c^* is chosen)P(at least one of the sample points contained in B_f)
= $\frac{1}{N_c}$[1- P(None of the points contained in B_f)]
= $\frac{1}{N_c}(1-(1-\frac{m(B_f(x^*,\varepsilon_f))}{m(c^*)})^{N_s})$

2. In the initialization stage we don't obtain c^*, but c^* is in the neighborhood of *icell*.
$P(X^t \in B_f(x^*, \varepsilon_f))$ = P(icell is in the first layer of neighborhood of c^*)P(at least one of the sample points contained in B_f)

>>

5. Final modified continuous tabu search: TS4

In this algorithm we allow more flexibility in size of neighborhood, i.e. a neighborhood can be enlarged or shrunk. We now introduce some new notations.

- <<tabu>> *tabu_list* = a list of (x^t, r^t) $k - L_T \le t \le k-1$, where k is the current iteration, and $r^t > 0$ is the size parameter.
- $R(x^t, r^t) = \{x : | x_i^t - x_i | \le r, i = 1, ..., n\}$ << r^t >> where n is the dimension of the search domain Ω.
- <<tabu>> $tabu\_region = \bigcup_{i=t-L_T}^{t-1} R(x^i, r^i)$

The parameters used in the implementation are as follow:

- $c > 1$, $\rho = .001$.
- $\Delta = \max_i (u_i - l_i)$, $M = \left\lfloor \frac{\ln \Delta - \ln \rho}{\ln c} \right\rfloor, m = \left\lfloor \frac{M}{2} \right\rfloor$, $r_{init} = 2^{1/m}$. <<why?>>
- The stopping criterion is the maximum number of iterations. For example, we use 10000.

Algorithm 4: TS4
1. Set up parameters $r_{init}, \rho, c > 1, \Delta = \max_i \{u_i - l_i\}$.
2. Set iteration counter *t=0, n=0.* << n → k since n =dim(x)>>
3. Randomly select a point x^0. <<use superscript t>>
4. $r^t = r_{init}$, $fbest = f(x^0)$, $y^0 = x^0$. <<use superscript t>>

5. Set *tabu_list* = $\varnothing$.
6. **Repeat**
flag =1.
While flag = 1 **do**
Call candidate3(x^t, r^t, N_s, y, *v*).
If *v* < *fbest* **then**
n=*n*+*1*, y^n= y, flag = 0.
else
If y $\notin$ *tabu_region* **then**
Enlarge the region $r^{t+1} = cr^t$.
If r^{t+1}>□ then $r^{t+1} = \rho$.
<<□ should be related to □, otherwise there might be a serious problem here, e.g. when □ > □ >>
Put y *and* r^{t+1} in *tabu_list*.
else
Shrink the region $r^{t+1} = r^t/c$.
If $r^{t+1} < \rho$ then $r^{t+1} = r_{init}$.
<<□ should be related to r_{init}, otherwise there might be a serious problem here, e.g. when □ > r_{init} >>
$y^{t+1} = y$.
<<This seems logically confusing and probably incorrect as well. y^n= y was used earlier! I don't see why you need two iteration counters t and n.>>
end if
end if
$x^{t+1} = y$.
Update *t* = *t* +*1*.
end while
until one of stopping criteria is satisfied.

Subroutine candidate3(z, r, M, y, *v*)
Input: z, r, M
Output: y, *v*
1. Randomly select M points from $\{ x : |z_i - x_i| \le r, i = 1,...,n\}$.
2. Choose the minimum among the *M* points and set it as y.
3. Set *v* = f(y).
4. Return y, *v*.

Since the sampled domain will be shrunk or enlarged, the probability of missing some nonempty set will be zero. The convergence proof can be shown similarly. First we need to approximate some probabilities.

- In the case of $B_f(x^*, \varepsilon_f) \cap R(x^t, r^t) = B_f(x^*, \varepsilon_f)$, the probability of selecting a point from $R(x^t, r^t)$ and obtaining a point in $B_f(x^*, \varepsilon_f)$ can be approximated as follows:

$$\frac{m(B_f(x^*,\varepsilon_f))}{m(\Omega)} \le \frac{m(B_f(x^*,\varepsilon_f))}{m(R(x^t,r^t))} \le \frac{m(B_f(x^*,\varepsilon_f))}{m(R(x^t,\rho))}$$. <<why?>>

The probability that we will obtain a neighborhood $R(x^t,r^t)$ in which $B_f(x^*,\varepsilon_f)\cap R(x^t,r^t)\neq\varnothing$ is approximated next. Recall that we defined

$$\eta = \frac{m(B_f(x^*,\varepsilon_f))}{m(\Omega)}.$$

$$P(B_f(x^*,\varepsilon_f)\cap R(x^t,r^t)\neq\varnothing) = P(B_f(x^*,\varepsilon_f)\cap R(x^t,r^t)\neq\varnothing \mid x^t\in B_f(x^*,\varepsilon_f)) + P(B_f(x^*,\varepsilon_f)\cap R(x^t,r^t)\neq\varnothing \mid x^t\notin B_f(x^*,\varepsilon_f))$$

<<Where is the randomness in these expressions?>>

$$= \frac{m(B_f(x^*,\varepsilon_f))}{m(R(x^t,r^t))}\cdot 1 + \ldots \quad \text{<<why?>>}$$

$$\ge \frac{m(B_f(x^*,\varepsilon_f))}{m(\Omega)} = \eta > 0 \ .$$

- P(At least one sample is contained in $B_f(x^*,\varepsilon_f)$)

  = 1 - P(None of the samples are contained in $B_f(x^*,\varepsilon_f)$)

  $$= 1-\left(1-\frac{m(B_f(x^*,\varepsilon_f))}{m(R(x^t,r^t))}\right)^{N_s}$$

  $$\ge 1-(1-\eta)^{N_s}$$

- The probability that the algorithm yields a point which belongs to the set $B_f(x^*,\varepsilon_f)$ is <<why?>> $P(B_f(x^*,\varepsilon_f)\cap R(x^t,r^t)\neq\varnothing)$ ×P(At least one sample is contained in $B_f(x^*,\varepsilon_f)$) $\ge \eta(1-(1-\eta)^{N_s}) = \gamma$.

The convergence depends on whether or not we have

$$R(x^t,r^t)\cap B_f(x^*,\varepsilon_f)\neq\varnothing,$$

which may occur in any iteration. Therefore,

$$P(X^t\in B_f(x^*,\varepsilon_f)) \ge \gamma+(1-\eta)\gamma+(1-\eta)^2\gamma+(1-\eta)^3\gamma+(1-\eta)^4\gamma+\ldots$$

<<explain?>>

$$= \gamma(1+(1-\eta)+(1-\eta)^2+(1-\eta)^3+(1-\eta)^4+\ldots)$$

$$= \frac{\gamma}{\eta} > 0 .$$

The rest is similar to Theorem 2.1. We also can show that $\frac{\gamma}{\eta}\ge\eta$ as in the next proposition.

Proposition 5.1 Let $\{x^t\}$ and $\{y^t\}$ be generated by Algorithm 3. Then

$$P(Y^t\in B_f(x^*,\varepsilon_f)) \ge P(X^t\in B_f(x^*,\varepsilon_f)) \ge \tilde{\square} \qquad (13)$$

Proof. Obviously, we only need to show

$$\frac{\gamma}{\eta} = 1-(1-\eta)^{N_s} \ge \eta .$$

We use induction on N_s.

$N_s = 1;\ 1-(1-\eta)=\eta$.

$N_s = 2;\ 1-(1-\eta)^2 = 2\eta-\eta^2 = \eta(2-\eta)\ge\eta$.

Suppose $1-(1-\eta)^{N_s} \geq \eta$.

Consider $1-(1-\eta)^{N_s+1} = 1-(1-\eta)^{N_s}(1-\eta)$

$= 1-(1-\eta)^{N_s} + \eta(1-\eta)^{N_s}$

$\geq \eta + \eta(1-\eta)^{N_s} = \eta(1+(1-\eta)^{N_s}) \geq \eta$. <<delete 2nd to the last>>

□

<<By now, our algs have been presented and analyzed. This is a good place to offer a discussion on differences between ours and existing continuous TS's. Point out our contribution.>>

6. Experimental results on tabu search

A number of standard test functions are chosen to show the effectiveness of our tabu search algorithms. Their optimal objective function values are available either through existing publications or by other reliable global optimization algorithms. The error in f* and the number of function evaluations (nfeval) are the averaged values taken over 100 runs. When algorithms 2 and 3 >are implemented, *ncell* is set to be 2 for 1-dimensional problem, 3 for 2-dimensional problem, and 4 for those problems with dimensions higher than 2. Tabu length L_T is assigned to be 10 for every run and promising length L_P of 3 is used if applicable. The number of samples in each cell (N_s) is set to be one for (TS1, TS2, TS3) <<What about TS4?>>. The stopping condition is the predetermined number of maximum iterations: 400 for n = 1, 800 for n = 2, and 1200 for the dimensions higher than 2.

Table 1 shows the average error and the number of function evaluations of three tabu search algorithms TS1, TS2, and TS3 with the same initial cell $(icell)$. TS3 performs best among them.

Table 2 shows the average error of f and number of function evaluations using algorithm 4 (TS4) and algorithm 3 (TS3) compared with the pure random search (PRS).

<<Comment on the missing entries in the tables.>>

From the experimental results we can draw the following conclusions.

1. All algorithms perform very well with problems of dimension one. For higher dimensional problems, much depends on the complexity of problem and the size of search domain.
2. All algorithms have difficulty when the search domain is large. Problems 18, 22, 27, 29, 35 have domains $[-100,100]\times[-100,100]$ and all algorithms did not perform well.
3. Algorithm 3 (TS3) and algorithm 4 (TS4) succeeded in more problems than the others. However, we must sacrifice the number of function evaluations. <<What if you let TS1, TS2, and PRS run longer, say reaching a compatible #(f)? Are TS3 and TS4 still better?>>
4. Algorithm 4 performs best, but with higher numbers of function evaluations. The reason could be that it allows the changes in the size of neighborhood. <<What if you let TS1, TS2, TS3, and PRS run longer, say reaching a compatible #(f)? Is TS4 still better?>>

7. Conclusion

Different types of neighborhoods have a direct impact on the performance of TS. The size of the neighborhood also has an effect on the convergence speed. The larger the neighborhood the faster the search domain is covered. However, we will need a larger sample size. We have tried several ways of defining neighborhoods. The results show that tabu search algorithm has advantage over pure random search if the design of neighborhood is efficient. However, as we try to reduce the search space by using tabu list or tabu region it may result in wasting the function evaluations without improving the best objective function value. The TS algorithms we implemented are based on the convergence of random search. The information about the gradient for guiding the moves has not been used.

References

[1] R. Battiti and G. Tecchiolli . The continuous reactive tabu search: blending combinatorial optimization and stochastic search for global optimization. *Annals of Operations Research*, 65:53-188,1996.

[2] R. Chelouah and P. Siarry. Tabu search applied to global optimization. *European Journal of Operational Research*, 123: 256-270, 2000.

[3] D. Cvijovic and J. Klinowski. *Handbook of Global Optimization*, Volume 2, Chapter 11 Taboo Search: An Approach to the Multiple-Minima Problem for Continuous Functions, pages 387-406. Kluwer Academic Publishers, 2002.

[4] F. Glover. Tabu search part I. *ORSA Journal on Computing*, 1:190-206, 1989.

[5] F. Glover and S. Hanafi. Tabu search and finite convergence. *Discrete Applied Mathematics*, 119(1-2):3-36, 2002.

[6] Francisco J. Solis and Roger J-B. Wets. Minimization by random search techniques. *Mathematics of Operation Research*, 6(1): 19-30, February 1981.

[7] Hanafi, S. 2001. the convergence of tabu search. Journal of Heuristics. Vol. 7, pp. 47 – 58.

[8] Ji, M. and Tang, H. 2004, Global optimizations and tabu search based on memory. *Applied Mathematics and Computation*. 2004. Vol. 159, pp. 449 – 457.

Table 1. Average error and average number of function evaluations for algorithms

TS1, TS2, and TS3 with the same initial point

		TS1		TS2		TS3	
#	dim	ave error	nfeval	ave error	nfeval	ave error	nfeval
1	1	1.00E-04	403	1.00E-05	1118	1.26E-04	460
2	1	1.10E-04	416	3.00E-05	1157	8.53E-05	393
3	1	1.15E-03	338	7.70E-04	783	1.51E-03	421
4	1	4.65E-04	380	5.30E-05	989	8.01E-04	393
5	1	5.56E-03	374	2.18E-03	781	5.56E-03	374
6	2	5.59E-02	424	1.55E-02	549	1.42E-03	5988
7	2	7.23E-02	534	9.01E-02	759	1.73E-03	6714
8	2	4.85E-02	386	1.57E-02	887	1.23E-03	8128
9	2	1.62E-02	301	6.27E-03	905	1.39E-04	6761
10	2	1.93E-01	254	1.20E-01	595	2.12E-03	6525
11	2	1.75E-01	308	1.75E-01	308	9.77E-03	5792
12	2	4.85E-01	548	1.71E-01	1445	3.79E-02	7227
13	2	1.48E-02	380	2.95E-03	1444	1.94E-04	5263
14	2	9.33E-02	381	4.19E-02	1198	7.26E-03	5260
15	2	7.16E-01	371	2.43E-01	861	1.04E-02	6197
16	2	7.10E-02	558	3.11E-02	562	3.31E-03	6755
17	2	8.26E-02	505	6.62E-02	909	9.87E-03	10071
18	2	2.89E+00	428	1.81E+00	882	8.35E-01	5361
19	2	6.36E-02	478	3.04E-02	896	7.13E-04	8078
20	2	2.18E-02	487	1.88E-02	694	9.93E-04	4410
21	2	6.00E-01	355	5.72E-01	764	5.47E-01	6397
22	2	9.99E-01	538	1.53E-01	399	1.53E-01	399
23	2	-	-	6.88E-01	159	3.85E-01	4774
24	2	-	-	6.02E-01	310	1.58E-01	5913
25	2	-	-	-	-	3.73E-02	28329
26	2	-	-	8.54E-01	378	2.55E-01	4873
27	2	-	-	9.64E-01	147	5.82E-01	7310
28	3	3.80E-02	6161	2.55E-01	605	3.80E-02	6162
29	3	-	-	-	-	8.94E-01	45587
30	3	6.53E-01	328	2.89E-01	650	1.75E-03	13782
31	3	9.84E-03	394	9.41E-03	747	1.23E-04	8512
32	4	-	-	-	-	8.11E-01	44492
33	5	4.67E-01	491	4.63E-01	317	1.83E-01	3790
34	6	4.79E-04	375	3.58E-04	1380	1.71E-06	102156
35	9	-	-	-	-	-	-
36	9	1.32E+00	213	9.11E-01	409	2.30E-03	2194543

Table 2. Average error of f and number of function evaluations using TS3, TS4, and pure random search (PRS)

		PRS		TS4		TS3	
#	dim	error	nfeval	error	nfeval	error	nfeval
1	1	8.02E-08	535	1.40E-07	5167	1.26E-04	460
2	1	6.65E-05	929	8.70E-07	1957	8.53E-05	393
3	1	3.71E-05	133	7.28E-05	1151	1.51E-03	421
4	1	2.42E-05	2401	6.09E-05	4957	8.01E-04	393
5	1	1.91E-05	19174	1.26E-05	8083	5.56E-03	374
6	2	6.72E-04	51189	7.49E-04	1531	1.42E-03	5988
7	2	3.53E-03	24359	2.17E-03	19603	1.73E-03	6714
8	2	5.89E-04	51189	1.70E-07	27711	1.23E-03	8128
9	2	3.76E-04	28294	3.04E-05	25759	1.39E-04	6761
10	2	3.78E-03	49686	1.89E-06	31489	2.12E-03	6525
11	2	5.34E-03	51189	9.33E-03	3829	9.77E-03	5792
12	2	1.19E-01	28691	4.34E-02	1923	3.79E-02	7227
13	2	7.47E-05	28204	2.13E-04	2539	1.94E-04	5263
14	2	3.60E-03	51189	7.40E-04	24795	7.26E-03	5260
15	2	5.62E-03	16927	2.20E-07	454305	1.04E-02	6197
16	2	9.20E-04	49174	6.86E-04	3433	3.31E-03	6755
17	2	9.72E-03	57563	9.85E-03	4577	9.87E-03	10071
18	2	9.54E-01	28294	7.23E-02	127881	8.35E-01	5361
19	2	2.59E-03	39616	6.10E-07	170613	7.13E-04	8078
20	2	1.52E-05	8069	5.78E-04	6995	9.93E-04	4410
21	2	5.46E-01	51919	5.49E-01	5141	5.47E-01	6397
22	2	6.76E-01	9154	3.55E-01	17703	1.53E-01	399
23	2	6.35E-01	24214	2.00E-07	138017	3.85E-01	4774
24	2	3.26E-01	51189	2.30E-07	113371	1.58E-01	5913
25	2	5.30E-01	28691	8.65E-02	1817	3.73E-02	28329
26	2	7.74E-02	16590	1.30E-07	208071	2.55E-01	4873
27	2	1.31E+00	28691	4.79E-01	14319	5.82E-01	7310
28	3	4.38E-02	18027	5.39E-04	116801	3.80E-02	6162
29	3	4.38E+00	1812	3.25E-03	318659	8.94E-01	45587
30	3	3.13E-02	21692	6.00E-08	105897	1.75E-03	13782
31	3	4.44E-05	24162	1.66E-06	246647	1.23E-04	8512
32	4	7.60E+01	14730	1.50E-03	359107	8.11E-01	44492
33	5	3.00E-01	67091	4.00E-01	2315	1.83E-01	3790
34	6	9.18E-05	23848	1.00E-08	38523	1.71E-06	102156
35	9	-	-	5.79E+01	233987	-	-
36	9	4.79E-01	45064	2.45E-05	1110731	2.30E-03	2194543

8. Testing functions

1. $f(x) = 5.0 + \sin(x) + \sin(\frac{10x}{3}) + \log(x) - 0.84x;$

search domain: $2.7 < x < 7.5$; $f(x^*) = 0.39869259$.

2. $f(x) = 2.0 + \sin(x) + \sin(\frac{2x}{3});$

search domain: $3.1 < x < 20.4$; $f(x^*) = 0.0940388$.

3. $f(x) = 6.0 + \sin(2x+1) + \sin(3x+2) + \sin(4x+3) + \sin(5x+4) + \sin(6x+5);$ <<same line>>

search domain: $-10 < x < 10$; $f(x^*) = 1.0899717$.

4. $f(x) = 1.0 + (x + \sin(x))\exp(-x^2);$

search domain: $-10 < x < 10$; $f(x^*) = 0.17576058$.

5. $f(x) = 0.02(12 + 3x - 3.5x^2 + 7.2x^3)(1 + \cos(4\pi x))(1 + 0.8\sin(3\pi x));$

search domain: $-1 < x < 1$; $f(x^*) = -.0679996$.

6. Rastrigin function

$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7;$

search domain: $-1 < x_j < 1,\ j = 1,2$; $x^* = (0,0)$; $f(x^*) = 0$.

7. Hump function

$f(x) = 1.03163 + 4.0x_1^2 - 2.1x_1^4 + \frac{x_1^6}{3} + x_1x_2 - 4.0(1.0 - x_2^2)x_2^2;$

search domain: $-5 < x_j < 5,\ j = 1,2$;

$x^* = (.0898 - .7126), (-.0898.7126)$; $f(x^*) = 0$.

8. $f(x) = 1.0x_1^2 + 0.2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.3;$

search domain: $-1 < x_j < 1,\ j = 1,2$; $x^* = (0,0)$; $f(x^*) = 0$.

9. $f(x) = 1.0x_1^2 + 2.0x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2) + 0.3;$

search domain: $-1 < x_j < 1,\ j = 1,2$; $x^* = (0,0)$; $f(x^*) = 0$.

10. $f(x) = \sum_{i=1}^{20} (x_1 + x_2 b_i + x_2^2 c_i - a_i);$ where

$a = (4.284, 4.149, 3.877, 0.533, 2.211, 2.389, 2.145, 3.231, 1.998, 1.379,$
$2.106, 1.428, 1.011, 2.179, 2.858, 1.388, 1.651, 1.593, 1.046, 2.152)$
$b = (.286, .973, .384, .276, .973, .543, .957, .948, .543, .797, .936, .889, .006,$
$.828, .399, .617, .939, .784, .072, .889)$
$c = (.645, .585, .310, .058, .455, .779, .259, .202, .028, .099, .142, .296, .175,$
$.180, .842, .039, .103, .620, .158, .704)$

search domain: $-10 < x_j < 10,\ j = 1,2$;

1 local minimum $f(2.35, -.319) = 20.9805$;

1 global minimum: $x^* = (.864, 1.23)$; $f(x^*) = 16.0817$.

11. $f(x) = x_1^2 + x_2^2 - \cos(18.0x_1) - \cos(18.0x_2) + 3.0;$

search domain: $-1 < x_j < 1,\ j = 1,2\ ;\ f(x^*) = 1.0$.

12. $f(x) = \sum_{i=1}^{2} \frac{x_i^2}{200} - \prod_{i=1}^{2} \cos(\frac{x_i}{\sqrt{i}}) + 2.0;$

search domain: $-100 < x_j < 100,\ j = 1,2\ ;\ f(x^*) = 1.0$.

13. $f(x) = (x_1 - 2)^2 + (x_2 - 2)^2;$

search domain: $-5 < x_j < 5,\ j = 1,2\ ;\ f(x^*) = 0.0$.

14. $f(x) = 1.0 + \sin^2(x_1) + \sin^2(x_2) - 0.1\exp(-x_1^2 - x_2^2);$

search domain: $-10 < x_j < 10,\ j = 1,2\ ;\ f(x^*) = 0.9$.

15. $f(x) = 100.0(x_2 - x_1^2)^2 + (6.4(x_2 - 0.5)^2 - x_1 - 0.6)^2;$

search domain: $-5 < x_j < 5,\ j = 1,2\ ;\ f(x^*) = 0$.

16. De Jong function

$f(x) = 100.0(x_1^2 - x_2)^2 + (1.0 - x_1)^2;$

search domain: $-2.048 < x_j < 2.048,\ j = 1,2\ ;\ x^* = (1,1)\ ;\ f(x^*) = 0$.

17. Schaffer function F6

$$f(x) = 0.5 + \frac{\sin^2(\sqrt{x_1^2 + x_2^2}) - 0.5}{(1.0 + .001(x_1^2 + x_2^2))^2};$$

search domain: $-100 < x_j < 100,\ j = 1,2\ ;\ x^* = (0,0)\ ;\ f(x^*) = 0$.

18. Schaffer function F7

$f(x) = (x_1^2 + x_2^2)^{0.25}[1.0 + \sin^2(50.0(x_1^2 + x_2^2)^{0.1})];$

search domain: $-100 < x_j < 100,\ j = 1,2\ ;\ x^* = (0,0)\ ;\ f(x^*) = 0$.

19. Branin RCOS

$$f(x) = (x_2 - \frac{5}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10;$$

search domain: $-5 < x_1 < 10, 0 < x_2 < 15$; no local minimum;

3 global minima: $x^* = (-\pi, 12.275), (\pi, 2.275), (9.42478, 2.475)$;

$f(x^*) = 0.397887$.

20. The six-hump camel back function

$$f(x) = (4.0 - 2.1x_1^2 + \frac{x_1^4}{3.0})x_1^2 + x_1x_2 + (-4.0 + 4.0x_2^2)x_2^2;$$

search domain: $-3 < x_1 < 3, -2 < x_2 < 2$;

$x^* = (-0.0898, 0.7126), (0.0898, -0.7126)$; $f(x^*) = -1.0316$.

21. Shubert

$f(x) = [\sum_{j=1}^{5} j\cos((j+1)x_1 + j)][\sum_{j=1}^{5} j\cos((j+1)x_2 + j)];$

search domain: $-10 < x_j < 10,\ j = 1,2$; 760 local minima; 18 global minima:

$f(x^*) = -186.7309$.

22. Easom

$f(x) = -\cos(x_1)\cos(x_2)\exp(-((x_1 - \pi)^2 + (x_2 - \pi)^2));$

search domain: $-100 < x_j < 100,\ j = 1,2$; several local minima (exact number

unspecified in usual literature); 1 global minimum: $x^* = (\pi, \pi)$; $f(x^*) = -1$.

23. Bohachevsky function #1

$$f(x) = x_1^2 + 2.0x_2^2 - 0.3\cos(3.0\pi x_1) - 0.4\cos(4.0\pi x_2) + 0.7;$$

search domain: $-50 < x_j < 50$, $j = 1,2$; $x^* = (0,0)$; $f(x^*) = 0$.

24. Bohachevsky function #2

$$f(x) = x_1^2 + 2.0x_2^2 - 0.3\cos(3.0\pi x_1)\cos(4.0\pi x_2) + 0.3;$$

search domain: $-50 < x_j < 50$, $j = 1,2$; $x^* = (0,0)$; $f(x^*) = 0$.

25. Bohachevsky function #3

$$f(x) = x_1^2 + 2.0x_2^2 - 0.3\cos(3.0\pi x_1) + \cos(4.0\pi x_2) + 0.3;$$

search domain: $-50 < x_j < 50$, $j = 1,2$; $x^* = (0,0)$; $f(x^*) = 0$.

26. Goldstein and Price

$$f(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 13x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$$
$$\cdot[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 - 48x_2 - 36x_1x_2 + 27x_2^2)];$$

search domain: $-2 < x_j < 2$, $j = 1,2$; 4 local minima;

1 global minimum: $x^* = (-1,0)$; $f(x^*) = 3$.

27. $f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7;$

search domain: $-100 < x_j < 100$, $j = 1,2$; several local minima (exact number unspecified in usual literature); 1 global minimum: $x^* = (0,0)$; $f(x^*) = 0$.

28. $f(x) = \sum_{i=1}^{3} |\min(d_i, x_i)|$; where

$b = (8, -0.4, 2)^T$, $A = \begin{bmatrix} 1 & -6 & -1 \\ 0 & 4 & -3 \\ -1 & -2 & 1 \end{bmatrix}$, $d = Ax + b$.

search domain: $-1 < x_j < 1$, $j = 1,2,3$; $x^* = (0, 0.1, 0)$; $f(x^*) = 0$.

29. Let $b_1 = (1,3,5)^T$, $b_2 = (2,1,3)^T$

$$A_1 = \begin{bmatrix} 1 & 0 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & 0 \\ 0 & -\frac{1}{2} & 1 \end{bmatrix}, \; A_2 = \begin{bmatrix} 1 & -\frac{1}{2} & 0 \\ -\frac{2}{3} & 1 & -\frac{1}{5} \\ -\frac{1}{4} & -\frac{1}{3} & 1 \end{bmatrix},$$

$u = A_1x - b_1$, $v = A_2x - b_2$.

$$f(x) = \sum_{i=1}^{3} |\max(u_i, v_i)|;$$

search domain: $-100 < x_j < 100$, $j = 1,2,3$;

$x^* = (3.7379, 4.5878, 5.4657)$; $f(x^*) = 0$.

30. De Jong (3 variables)

$f(x) = x_1^2 + x_2^2 + x_3^2$;

search domain: $-5.12 < x_j < 5.12$, $j = 1,2,3$; 1 single minimum (global):

$x^* = (0,0,0)$; $f(x^*) = 0$.

31. Hartmann

$$f(x) = -\sum_{i=1}^{4} c_i \exp[-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2];$$

search domain: $0 < x_j < 1,\ j = 1,2,3$; 4 local minima:

$p_i = (p_{i1}, p_{i2}, p_{i3}) = i$ th local minimum approximation; $f(p_i) \cong -c_i$;

1 global minimum: $x^* = (0.11, 0.555, 0.855)$; $f(x^*) = -3.86278$.

i	a_{ij}			c_i	P_{ij}		
1	3.0	10.0	30.0	1.0	0.3689	0.1170	0.2673
2	0.1	10.0	35.0	1.2	0.4699	0.4387	0.7470
3	3.0	10.0	30.0	3.0	0.1091	0.8732	0.5547
4	0.1	10.0	35.0	3.2	0.0381	0.5743	0.8828

32. The Colville function

$$f(x) = 100.0(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2$$
$$+10.1(x_2 - 1)^2 + (x_4 - 1)^2 + 19.8(x_2 - 1)(x_4 - 1);$$

search domain: $-10 < x_j < 10,\ j = 1,...,4$; $x^* = (1,1,1,1)$; $f(x^*) = 0$.

33. De Jong function F3

$$f(x) = \sum_{j=1}^{5} \lfloor x_j \rfloor;$$

search domain: $-5.12 < x_j < 5.12,\ j = 1,...,5$;

$f(x^*) = -30$ for all $-5.12 \le x_j \le -5.0$.

34. Griewank's function

$$f(x) = \frac{1}{600}\sum_{j=1}^{6} x_j^2 - \prod_{j=1}^{6} \frac{\cos(x_j)}{j} + \frac{1}{720};$$

search domain: $-1 < x_j < 1,\ j = 1,...,6$; $x^* = (0,0,0,0,0,0)$; $f(x^*) = 0$.

35. Rosenbrock's function (9 variables)

$$f(x) = \sum_{i=1}^{8} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2;$$

search domain: $-100 < x_j < 100,\ j = 1,...,9$; $x^* = (1,1,...,1)$; $f(x^*) = 0$.

36. Zakharov's function (9 variables)

$$f(x) = \sum_{i=1}^{9} x_i^2 + (\sum_{i=1}^{9} 0.5ix_i)^2 + (\sum_{i=1}^{9} 0.5ix_i)^4;$$

search domain: $-1 < x_j < 1,\ j = 1,...,9$; $x^* = (0,...,0)$; $f(x^*) = 0$.