
Computational Optimization and Applications, 31, 87–115, 2005

Discrete Filled Function Method for Discrete Global Optimization

CHI-KONG NG ckng@se.cuhk.edu.hk
Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong S.A.R., People's Republic of China

LIAN-SHENG ZHANG
Department of Mathematics, Shanghai University, Baoshan, Shanghai 200436, People's Republic of China

DUAN LI* dli@se.cuhk.edu.hk
Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong S.A.R., People's Republic of China

WEI-WEN TIAN wwtian@guomai.sh.cn
Department of Mathematics, Shanghai University, Baoshan, Shanghai 200436, People's Republic of China

Received November 25, 2002; Revised April 21, 2004; Accepted June 11, 2004

Abstract. A discrete filled function method is developed in this paper to solve discrete global optimization problems over "strictly pathwise connected domains." Theoretical properties of the proposed discrete filled function are investigated and a solution algorithm is proposed. Numerical experiments reported in this paper on several test problems with up to 200 variables have demonstrated the applicability and efficiency of the proposed method.

Keywords: nonlinear integer programming, quadratic integer programming, linear integer programming, discrete global optimization, discrete filled function method

1. Introduction

Optimization of a general cost function over discrete variables arises frequently in various applications such as combinatorics, scheduling, design and operations problems. Even though some discrete optimization problems have been studied since ancient times, the impacts of discrete optimization have become influential only in the last few decades due to the advancement of computer technologies. The literature on discrete global optimization can be divided into two classes, deterministic approaches and stochastic approaches.

Various deterministic solution methods for solving discrete optimization problems have been proposed during the last two decades. The branch and bound method (see, e.g., [12, 15]) searches a solution based on successively subdividing the feasible region and estimating lower bounds by solving the continuous relaxation problems. It is clear that branch

*Author to whom correspondence should be addressed.

and bound methods may fail to find an optimal solution in the absence of convexity. When the objective function and constraint functions are separable and there exist only a few constraints, dynamic programming method (see, e.g., [3, 4]) can be used to solve the discrete optimization problem. Lagrangian method (see, e.g., [6, 10]) deals with discrete optimization problems in an indirect way by incorporating constraints into the objective function, forming a relaxation problem and performing a dual search to find an optimal primal solution. Unlike its success in continuous optimization, Lagrangian method often fails to identify an optimal primal solution even in linear or convex situations. Nonlinear Lagrangian formulations [17, 18, 26, 28] have been recently proposed to guarantee an existence of an optimal primal-dual pair and an identification of an optimal primal solution via dual search. A major difficulty associated with nonlinear Lagrangian formulations is how to globally and efficiently solve the resulting nonlinear Lagrangian relaxation problem.

Stochastic approaches for discrete optimization include, for example, the simulated annealing algorithms, the genetic algorithms, and the tabu search algorithms. Recently, Litinetski and Abramzon [19] generalized the existing random search methods with systematic reduction of the search region (see, e.g., [21]) and suggested a new multi-start, adaptive, random search method; Mohan and Nguyen [22] modified the traditional controlled random search technique (see, e.g., [24]) and incorporated the simulated annealing concept of accepting occasional uphill moves to achieve a higher reliability in obtaining a global optimal solution. Although there is no guarantee of attaining an optimality when adopting stochastic methods, they are quite efficient in finding a near-optimal solution for a variety of discrete global optimization problems.

Several continuous global optimization methods, such as the tunneling algorithm [16] and the filled function method [7], search for a global minimum among the local minima. More specifically, they invoke certain auxiliary functions to move successively from one local minimum to another better one. The concept of the filled functions was introduced by Ge in [7] for continuous global optimization. Thereafter further results on filled function methods have been reported by various authors (see, e.g., [9, 13, 20, 27, 30]). In order to tackle discrete global optimization problems, Ge and Huang [8] and Zhang et al. [29] transformed discrete global optimization problems into continuous global optimization problems and then solve them by the continuous filled function method [7] or some other continuous global optimization algorithms. Zhu [32] also used a continuous filled function method to solve discrete global optimization problems, while using a discrete direct search method to obtain a local minimum. In general, the third condition of the continuous filled function does not hold in discrete cases. (The details are discussed in the next section.) Difficulties may also occur when applying continuation methods to discrete optimization problems where the gradient vectors are unavailable or expensive to compute, as indicated in [29]. It is clear that the implementability and efficiency of the existing solution schemes of applying continuous filled function methods to solve discrete optimization problems need further improvement.

The main purpose of this paper is to propose and formalize a discrete version of the filled functions. By exploring certain special features of integer programming, the proposed discrete filled function seems to be better suited for discrete global optimization, as witnessed in our numerical experiments. The paper is organized as follows. Following this introduction, we present some preliminaries in Section 2 and define then a discrete filled function. In

Section 3, we propose a new discrete filled function and investigate its properties as well. In Section 4, we consider the numerical implementation of the proposed discrete filled function and suggest a solution algorithm. In Section 5, we first demonstrate the solution procedures of the algorithm by an illustrative example. We then report the results of the algorithm in solving several test problems with up to 200 variables. Finally, we draw some conclusions in Section 6.

2. Preliminaries

Let $\mathbb{Z}^n$ be the set of integer points in $\mathbb{R}^n$. We consider the following discrete global optimization problem:

$$(P) \quad \min_{x \in X \subset \mathbb{Z}^n} f(x)$$

and make the following assumptions in this paper:

Assumption 1. $X \subset \mathbb{Z}^n$ is a bounded set which contains more than one point. This implies that there exists a constant $K > 0$ such that

$$1 \leq K = \max_{x^{(1)}, x^{(2)} \in X} \left\| x^{(1)} - x^{(2)} \right\| < \infty, \tag{1}$$

where $\|\cdot\|$ is the usual Euclidean norm.

Assumption 2. X is a strictly pathwise connected domain (see Definition 1).

Assumption 3. $f: X \mapsto \mathbb{R}$ satisfies the following Lipschitz condition for every $x^{(1)}, x^{(2)} \in X$:

$$\left| f\left(x^{(1)}\right) - f\left(x^{(2)}\right) \right| \leq L \left\| x^{(1)} - x^{(2)} \right\|, \tag{2}$$

where $0 < L < \infty$ is a constant.

Notice that the formulation in (P) allows the set X to be defined by box constraints as well as inequality constraints. Furthermore, when f is coercive, i.e., $f \to \infty$ as $\|x\| \to \infty$, there always exists a box which contains all discrete global minimizers of f. Thus, the unconstrained discrete global optimization problem, $\min_{x \in \mathbb{Z}^n} f(x)$ can be reduced into an equivalent problem formulation in (P). In other words, both unconstrained and constrained discrete global optimization problems can be considered in (P).

To simplify the discussion in this paper, we recall some definitions in discrete analysis and discrete optimization. Moreover, we extend some definitions in continuous global optimization to discrete global optimization.

Definition 1. A sequence $\{x^{(i)}\}_{i=-1}^{u}$ is called a DISCRETE PATH in X between two distinct points x^* and x^{**} in X if $x^{(-1)} = x^*$, $x^{(u)} = x^{**}$, $x^{(i)} \in X$, for all i; $x^{(i)} \neq x^{(j)}$, for $i \neq j$; and $\|x^{(0)} - x^*\| = \|x^{(i+1)} - x^{(i)}\| = \|x^{**} - x^{(u-1)}\| = 1$, for all i. If, in addition, $\|x^{(i)} - x^*\| < \|x^{(i+1)} - x^*\|$, for all i, or equivalently, $\|x^{(i)} - x^{**}\| > \|x^{(i+1)} - x^{**}\|$, for all

i, the sequence is also called a STRICT DISCRETE PATH in X between x^* and x^{**}. If such a (strict) discrete path exists, then x^* and x^{**} are said to be (STRICTLY) PATHWISE CONNECTED in X. Furthermore, if every two distinct points in X are (strictly) pathwise connected in X, then X is called a (STRICTLY) PATHWISE CONNECTED DOMAIN.

Example 1. Let $X = \{x \in \Omega \cap \mathbb{Z}^n : \Omega \subset \mathbb{R}^n \text{ is convex}\}$. If X is a nonempty and pathwise connected domain, then X is a strictly pathwise connected domain.

Example 2. Let $\Gamma = \{(x_1, x_2) \in \mathbb{R}^2 : x_1 \geq 0,\ x_2 \leq 2,\ x_2 \leq x_1^2\}$. Then Γ is a nonconvex domain but $X = \Gamma \cap \mathbb{Z}^2$ is a strictly pathwise connected domain.

Definition 2. The set of all directions in $\mathbb{Z}^n$ is defined by $D = \{\pm e_i : i = 1, 2, \ldots, n\}$, where e_i is the ith unit vector (the n dimensional vector with the ith component equal to one and all other components equal to zero).

Definition 3. For any $x \in \mathbb{Z}^n$, the DISCRETE NEIGHBORHOOD of x is defined by $N(x) = \{x,\ x \pm e_i : i = 1, 2, \ldots, n\}$.

Definition 4. The DISCRETE INTERIOR of X is defined by $\operatorname{int} X = \{x \in X : N(x) \subset X\}$. While, the DISCRETE BOUNDARY of X is denoted by $\partial X = X \setminus \operatorname{int} X$.

Note that, if X contains less than or equal to $2n$ points, then $\operatorname{int} X = \emptyset$ and $\partial X = X$.

Definition 5. A point $x^* \in X$ is called a DISCRETE LOCAL MINIMIZER of f over X if $f(x^*) \leq f(x)$, for all $x \in X \cap N(x^*)$. Furthermore, if $f(x^*) \leq f(x)$, for all $x \in X$, then x^* is called a DISCRETE GLOBAL MINIMIZER of f over X. If, in addition, $f(x^*) < f(x)$, for all $x \in X \cap N(x^*) \setminus x^*$, then x^* is called a STRICT DISCRETE LOCAL (GLOBAL) MINIMIZER of f over X.

Definition 6. A point $\tilde{x} \in X$ is called a DISCRETE SADDLE POINT of f over X if there exists a partition $\{A, B\}$ of the set $\{1, 2, \ldots, n\}$ such that $A, B \neq \emptyset$; $A \cap B = \emptyset$; $A \cup B = \{1, 2, \ldots, n\}$; $f(\tilde{x} \pm e_i) \geq f(\tilde{x})$, for all $i \in A$; and $f(\tilde{x} \pm e_j) \leq f(\tilde{x})$, for all $j \in B$. Furthermore, if $f(\tilde{x} \pm e_i) > f(\tilde{x})$, for all $i \in A$, and $f(\tilde{x} \pm e_j) < f(\tilde{x})$, for all $j \in B$, then $\tilde{x}$ is called a STRICT DISCRETE SADDLE POINT of f over X.

Definition 7. For any $x \in X, d \in D$ is said to be a DESCENT DIRECTION of f at x over X if $x + d \in X$ and $f(x + d) < f(x)$; besides, $d^* \in D$ is called a DISCRETE STEEPEST DESCENT DIRECTION of f at x over X if $f(x + d^*) \leq f(x + d)$, for all $d \in D^*$, where D^* is the set of all descent directions of f at x over X.

In the following, we present a discrete steepest descent method for finding a local minimizer of f over X from a given initial point $x \in X$.

Algorithm 1 (Discrete steepest descent method)

1. *Start from an initial point $x \in X$.*

2. *If x is a local minimizer of f over X, then stop. Otherwise, a discrete steepest descent direction d^* of f at x over X can be found.*
3. *Let $x := x + \lambda d^*$, where $\lambda \in \mathbb{Z}_+$ is the step length such that f has maximum reduction in the direction d^*, and go to Step* 2.

In the context of continuous global optimization, we have definitions of a basin, a lower/higher basin, a hill, etc. [5, 7]. We define similar terms in the following for discrete global optimization.

Definition 8. Given that x^* is a local minimizer of f over X. B^* is said to be a DISCRETE BASIN of f at x^* over X if $B^* \subset X$ is a pathwise connected domain which contains x^* and in which a discrete steepest descent trajectory from any initial point in B^* converges to x^*, but outside which the discrete steepest descent trajectory from any initial point in $X \backslash B^*$ does not converge to x^*.

Definition 9. Given that x^* and x^{**} are two distinct local minimizers of f over X. The discrete basin B^{**} of f at x^{**} over X is said to be LOWER than the discrete basin B^* of f at x^* over X if $f(x^{**}) < f(x^*)$. Equivalently, B^* is said to be HIGHER than B^{**}.

Definition 10. Given that $\hat{x}$ is a local minimizer of $-f$. The discrete basin of $-f$ at $\hat{x}$ over X is called the DISCRETE HILL of f at $\hat{x}$ over X.

The continuous filled function was introduced by Ge [7] as follows:

Definition 11 (Continuous filled function in [7]). Given that ${x_c}^*$ is an isolated local minimizer of $f\colon X_c \mapsto \mathbb{R}$, where $X_c \subset \mathbb{R}^n$. Let ${B_c}^*$ be the basin of f at ${x_c}^*$ over X_c. A function $F\colon X_c \mapsto \mathbb{R}$ is said to be a FILLED FUNCTION of f at ${x_c}^*$ if it satisfies the following:

(C1) ${x_c}^*$ is a maximizer of F and the whole basin ${B_c}^*$ of f at ${x_c}^*$ over X_c becomes a part of a hill of F;
(C2) F has no minimizers or saddle points in any basin of f higher than ${B_c}^*$;
(C3) if f has a basin ${B_c}^{**}$ at ${x_c}^{**}$ that is lower than ${B_c}^*$, then there is a point ${x_c}' \in {B_c}^{**}$ that minimizes F on the line through ${x_c}^*$ and the current iterative point.

We now modify it for discrete cases.

Definition 12 (Discrete filled function). Given that x^* is a discrete local minimizer of $f\colon X \mapsto \mathbb{R}$, where $X \subset \mathbb{Z}^n$. Let B^* be the discrete basin of f at x^* over X. A function $F\colon X \mapsto \mathbb{R}$ is said to be a DISCRETE FILLED FUNCTION of f at x^* if it satisfies the following:

(D1) x^* is a strict local maximizer of F over X;
(D2) F has no discrete local minimizers in B^* or in any discrete basin of f higher than B^*;
(D3) if f has a discrete basin B^{**} at x^{**} that is lower than B^*, then there is a discrete point $x' \in B^{**}$ that minimizes F on a discrete path $\{x^*, \ldots, x', \ldots, x^{**}\}$ in X.

Adopting the concept of the filled functions in [7], we solve a discrete global optimization problem via a two-phase cycle. In Phase 1 (Local Search), we start from a given point in X and use the discrete steepest descent method (Algorithm 1) to find a local minimizer x^* of f over X. In Phase 2 (Global Search), we construct a discrete filled function at x^* and search for a minimum of the discrete filled function on a discrete path in X via some special search directions, in order to identify a point x' in a discrete basin lower than the discrete basin of f at x^*. This process repeats until the time when minimizing the discrete filled function does not yield a better solution. The current local minimizer will be then taken as a global minimizer of f.

Notice that, a continuous filled function requires that there be no saddle points in any basin of f higher than the current basin, otherwise the saddle points in higher basins may stop the minimization process in Phase 2. On the contrary, a discrete saddle point of f does have a descent direction of f over X provided it is not a local minimizer of f at the same time. Hence the existence of discrete saddle points in discrete basins of f higher than the current discrete basin will not stop the minimization process in Phase 2.

A continuous filled function requires that if f has a basin $B_c{}^{**}$ that is lower than the current basin $B_c{}^*$, then there is a point $x_c{}'$ in $B_c{}^{**}$ that minimizes the continuous filled function on a straight line connecting $x_c{}^*$ and the current iterative point x. These requirements are unlikely to be satisfied in discrete cases, not only because $x_c{}'$ may not be a discrete point, but also because not every point on the straight line is feasible in a discrete domain. In the proposed discrete filled function method, we consider minimization on a discrete path that in general is not a straight line.

3. Discrete filled function and its properties

Now, we propose a two-parameter discrete filled function for problem (P) at a local minimizer x^* as follows:

$$F_{\mu,\rho}(x; x^*) = f(x^*) - \min[f(x^*), f(x)] - \rho \|x - x^*\|^2 + \mu\{\max[0, f(x) - f(x^*)]\}^2. \tag{3}$$

In other words, when $f(x) \geq f(x^*)$,

$$F_{\mu,\rho}(x; x^*) = \mu [f(x) - f(x^*)]^2 - \rho \|x - x^*\|^2, \tag{4}$$

while when $f(x) \leq f(x^*)$,

$$F_{\mu,\rho}(x; x^*) = f(x^*) - f(x) - \rho \|x - x^*\|^2. \tag{5}$$

In the following subsections, we will show that $F_{\mu,\rho}(\cdot; x^*)$ satisfies the conditions to be qualified as a discrete filled function under certain conditions on the parameters ρ and μ.

3.1. The proposed discrete filled function satisfies condition (D1)

Lemma 1. *Given that x^* is a local minimizer of f over X. Suppose that $\bar{x} \in X$ is a point such that $\bar{x} \neq x^*$ and $f(\bar{x}) \geq f(x^*)$. If $\rho > 0$ and $0 \leq \mu < \rho/L^2$, then $F_{\mu,\rho}(\bar{x}; x^*) < 0 = F_{\mu,\rho}(x^*; x^*)$.*

Proof: Since $f(\bar{x}) \geq f(x^*)$, then, by Assumption 3, we have

$$0 \leq f(\bar{x}) - f(x^*) \leq L \|\bar{x} - x^*\|.$$

Since $\|\bar{x} - x^*\| > 0$, so if $\rho > 0$ and $0 < \mu < \rho/L^2$, then

$$\begin{aligned} F_{\mu,\rho}(\bar{x}; x^*) &= \mu [f(\bar{x}) - f(x^*)]^2 - \rho \|\bar{x} - x^*\|^2 \\ &\leq \mu L^2 \|\bar{x} - x^*\|^2 - \rho \|\bar{x} - x^*\|^2 < 0 = F_{\mu,\rho}(x^*; x^*). \end{aligned}$$

This completes the proof. □

Corollary 2. *Given that x^* is a local minimizer of f over X. If $\rho > 0$ and $0 \leq \mu < \rho/L^2$, then x^* is a strict local maximizer of $F_{\mu,\rho}(\cdot; x^*)$ over X. If, in addition, x^* is a global minimizer of f over X, then $F_{\mu,\rho}(x; x^*) < 0$, for all $x \in X \setminus x^*$.*

Proof: Since x^* is a local minimizer of f over X, thus $f(\bar{x}) \geq f(x^*)$, for all $\bar{x} \in X \cap N(x^*)$. Hence, by Lemma 1, $F_{\mu,\rho}(\bar{x}; x^*) < 0 = F_{\mu,\rho}(x^*; x^*)$, for all $\bar{x} \in X \cap N(x^*) \setminus x^*$. Thus, x^* is a strict local maximizer of $F_{\mu,\rho}(\cdot; x^*)$.

If x^* is a global minimizer of f over X, then $f(x) \geq f(x^*)$, for all $x \in X$. Thus, by Lemma 1, the result follows. □

From Corollary 2, we conclude that $F_{\mu,\rho}(\cdot; x^*)$ satisfies the discrete filled function condition (D1) if $\rho > 0$ and $0 \leq \mu < \rho/L^2$.

3.2. The proposed discrete filled function satisfies condition (D2)

Lemma 3. *Given that $x^{(1)}$, $x^{(2)}$, and x^* are three distinct points in X. If $\|x^{(2)} - x^*\| > \|x^{(1)} - x^*\|$, then*

$$1 \leq \frac{\left\|x^{(2)} - x^{(1)}\right\|}{\left\|x^{(2)} - x^*\right\| - \left\|x^{(1)} - x^*\right\|} < 2K^2.$$

Proof: On one hand, we consider

$$\left\|x^{(2)} - x^*\right\| \leq \left\|x^{(2)} - x^{(1)}\right\| + \left\|x^{(1)} - x^*\right\|.$$

Since $\|x^{(2)} - x^*\| > \|x^{(1)} - x^*\|$, we have

$$0 < \left\|x^{(2)} - x^*\right\| - \left\|x^{(1)} - x^*\right\| \leq \left\|x^{(2)} - x^{(1)}\right\|,$$

and hence

$$1 \leq \frac{\left\|x^{(2)} - x^{(1)}\right\|}{\left\|x^{(2)} - x^*\right\| - \left\|x^{(1)} - x^*\right\|}.$$

On the other hand, we consider for every $x \in \mathbb{Z}^n$, $\|x\|^2 \in \mathbb{Z}$ and hence $\|x^{(2)} - x^*\|^2 - \|x^{(1)} - x^*\|^2 \in \mathbb{Z}$. Since $\|x^{(2)} - x^*\| > \|x^{(1)} - x^*\|$, we have

$$\left\|x^{(2)} - x^*\right\|^2 - \left\|x^{(1)} - x^*\right\|^2 \geq 1.$$

Moreover, due to $\|x^{(2)} - x^*\| > \|x^{(1)} - x^*\|$, we have

$$\left\|x^{(2)} - x^*\right\| + \left\|x^{(1)} - x^*\right\| > 0.$$

Therefore, from Assumption 1, we have

$$\left\|x^{(2)} - x^*\right\| - \left\|x^{(1)} - x^*\right\| \geq \frac{1}{\left\|x^{(2)} - x^*\right\| + \left\|x^{(1)} - x^*\right\|} > \frac{1}{2K},$$

and thus

$$\frac{\left\|x^{(2)} - x^{(1)}\right\|}{\left\|x^{(2)} - x^*\right\| - \left\|x^{(1)} - x^*\right\|} < \frac{K}{1/(2K)} = 2K^2.$$

This completes the proof. □

Lemma 4. *Given that x^* is a local minimizer of f over X. Suppose that $x^{(1)}, x^{(2)} \in X$ are two points such that $0 < \|x^{(1)} - x^*\| < \|x^{(2)} - x^*\|$ and $f(x^*) \leq f(x^{(1)}) \leq f(x^{(2)})$. If $\rho > 0$ and $0 \leq \mu \leq \rho/(2K^2L^2)$, then*

$$F_{\mu,\rho}\left(x^{(2)}; x^*\right) < F_{\mu,\rho}\left(x^{(1)}; x^*\right) < 0 = F_{\mu,\rho}(x^*; x^*).$$

Proof: Consider

$$\begin{aligned} &F_{\mu,\rho}\left(x^{(2)}; x^*\right) - F_{\mu,\rho}\left(x^{(1)}; x^*\right) \\ &\quad = \mu\left\{\left[f\left(x^{(2)}\right) - f(x^*)\right]^2 - \left[f\left(x^{(1)}\right) - f(x^*)\right]^2\right\} \\ &\qquad - \rho\left(\left\|x^{(2)} - x^*\right\|^2 - \left\|x^{(1)} - x^*\right\|^2\right) \end{aligned}$$

$$
\begin{aligned}
&= \left(\left\|x^{(2)} - x^*\right\|^2 - \left\|x^{(1)} - x^*\right\|^2\right) \\
&\quad \cdot \left\{ \mu \frac{\left[f\left(x^{(2)}\right) - f(x^*)\right]^2 - \left[f\left(x^{(1)}\right) - f(x^*)\right]^2}{\left\|x^{(2)} - x^*\right\|^2 - \left\|x^{(1)} - x^*\right\|^2} - \rho \right\} \\
&= \left(\left\|x^{(2)} - x^*\right\|^2 - \left\|x^{(1)} - x^*\right\|^2\right) \\
&\quad \cdot \left\{ \mu \left[\frac{f\left(x^{(2)}\right) - f(x^*) + f\left(x^{(1)}\right) - f(x^*)}{\left\|x^{(2)} - x^*\right\| + \left\|x^{(1)} - x^*\right\|} \right] \right. \\
&\quad \left. \cdot \left[\frac{f\left(x^{(2)}\right) - f\left(x^{(1)}\right)}{\left\|x^{(2)} - x^*\right\| - \left\|x^{(1)} - x^*\right\|} \right] - \rho \right\}.
\end{aligned}
$$

Then, by Assumption 3 and Lemma 3, we have

$$
\begin{aligned}
&F_{\mu,\rho}\left(x^{(2)}; x^*\right) - F_{\mu,\rho}\left(x^{(1)}; x^*\right) \\
&< \left(\left\|x^{(2)} - x^*\right\|^2 - \left\|x^{(1)} - x^*\right\|^2\right)(\mu \cdot L \cdot 2K^2L - \rho) \le 0.
\end{aligned}
$$

The second inequality in the conclusion of the lemma follows directly from Lemma 1. □

Lemma 5. *Given that x^* is a local minimizer of f over X. Suppose that $x^{(1)}, x^{(2)} \in X$ are two points such that $0 < \|x^{(1)} - x^*\| < \|x^{(2)} - x^*\|$ and $f(x^*) \le f(x^{(2)}) \le f(x^{(1)})$. If $\rho > 0$ and $\mu \ge 0$ then*

$$
F_{\mu,\rho}\left(x^{(2)}; x^*\right) < F_{\mu,\rho}\left(x^{(1)}; x^*\right) < 0 = F_{\mu,\rho}(x^*; x^*).
$$

Proof: Consider

$$
\begin{aligned}
&F_{\mu,\rho}\left(x^{(2)}; x^*\right) - F_{\mu,\rho}\left(x^{(1)}; x^*\right) \\
&\quad = \mu \left\{ \left[f\left(x^{(2)}\right) - f(x^*)\right]^2 - \left[f\left(x^{(1)}\right) - f(x^*)\right]^2 \right\} \\
&\qquad - \rho\left(\left\|x^{(2)} - x^*\right\|^2 - \left\|x^{(1)} - x^*\right\|^2\right) < 0.
\end{aligned}
$$

The second inequality in the conclusion of the lemma follows directly from Lemma 1. □

Theorem 6. *Given that x^* is a local minimizer of f over X. Suppose that $\bar{x} \in X$ is a point such that $\bar{x} \ne x^*$ and $f(\bar{x}) \ge f(x^*)$. If $\rho > 0$ and $0 \le \mu \le \rho/(2K^2L^2)$, then for each $\bar{d} \in D$ such that $\bar{x} + \bar{d} \in X$, $f(\bar{x} + \bar{d}) \ge f(x^*)$ and $\|\bar{x} + \bar{d} - x^*\| > \|\bar{x} - x^*\|$, it satisfies*

$$
F_{\mu,\rho}(\bar{x} + \bar{d}; x^*) < F_{\mu,\rho}(\bar{x}; x^*) < 0 = F_{\mu,\rho}(x^*; x^*). \tag{6}
$$

Proof: Consider the following two cases:

1. If $f(\bar{x}) \le f(\bar{x} + \bar{d})$, we have $f(x^*) \le f(\bar{x}) \le f(\bar{x} + \bar{d})$. Due to $0 < \|\bar{x} - x^*\| < \|\bar{x} + \bar{d} - x^*\|$, $\rho > 0$ and $0 \le \mu \le \rho/(2K^2L^2)$, then (6) holds by Lemma 4.

2. If $f(\bar{x}) \geq f(\bar{x}+\bar{d})$, we have $f(x^*) \leq f(\bar{x}+\bar{d}) \leq f(\bar{x})$. Due to $0 < \|\bar{x}-x^*\| < \|\bar{x}+\bar{d}-x^*\|$, $\rho > 0$ and $\mu \geq 0$, then (6) holds by Lemma 5.

This completes the proof. □

Theorem 6 clearly reveals that $\bar{d}$ is a descent direction of $F_{\mu,\rho}(\bar{x};x^*)$ if $f(\bar{x}+\bar{d}) \geq f(x^*)$. Therefore, $F_{\mu,\rho}(\cdot;x^*)$ satisfies the discrete filled function condition (D2) if $\rho > 0$ and $0 \leq \mu \leq \rho/(2K^2L^2)$. We further enhance this conclusion by the following corollary.

Corollary 7. *Given that x^* is a local minimizer of f over X and B^* is the discrete basin of f at x^*. If $\rho > 0$ and $0 \leq \mu \leq \rho/(2K^2L^2)$, then $F_{\mu,\rho}(\cdot;x^*)$ has no local minimizers in B^* or in any discrete basin of f higher than B^*.*

Proof: From the definition of the discrete basin (Definition 8), every point $\bar{x}$ in B^* or in any discrete basin of f higher than B^*, then $f(\bar{x}+\bar{d}) \geq f(x^*)$, for all $\bar{d} \in \{d \in D: \bar{x}+d \in X, \|\bar{x}+d-x^*\| > \|\bar{x}-x^*\|\}$. Therefore, from Theorem 6, we have $F_{\mu,\rho}(\bar{x}+\bar{d};x^*) < F_{\mu,\rho}(\bar{x};x^*) < 0 = F_{\mu,\rho}(x^*;x^*)$. This completes the proof. □

3.3. The proposed discrete filled function satisfies condition (D3)

Theorem 8. *Given that x^* and x^{**} are two distinct local minimizers of f over X such that $f(x^*) > f(x^{**})$. Let x' and $x'+d'$ be two points in the discrete basin B^{**} of f at x^{**} over X, where d' is a descent direction of f at x' over X and*

$$f(x') \geq f(x^*) > f(x'+d'). \tag{7}$$

Suppose that there is a strict discrete path $\{x^{(i)}\}_{i=-1}^{u}$ in X between $x^*(\equiv x^{(-1)})$ and $x'(\equiv x^{(u)})$ such that $f(x^{(i)}) \geq f(x^*)$, for all $i = 0, 1, 2, \ldots, u$. If $0 \leq \mu \leq \rho/(2K^2L^2)$ and $0 < \rho < \rho_0$, where

$$\rho_0 = \frac{f(x^*) - f(x'+d')}{\|x'+d'-x^*\|^2}, \tag{8}$$

then there is a discrete path $\{x^{(i)}\}_{i=-1}^{u+v}$ in X between $x^*(\equiv x^{(-1)})$ and $x^{**}(\equiv x^{(u+v)})$ such that $x'+d' \equiv x^{(u+1)}$, $x' \equiv x^{(u)}$ and x' is a minimizer of $F_{\mu,\rho}(\cdot;x^*)$ on the discrete path. If, in addition,

$$0 < \rho < \min\left\{\rho_0,\ \frac{f\left(x^{(u+j-1)}\right) - f\left(x^{(u+j)}\right)}{\left\|x^{(u+j)}-x^*\right\|^2 - \left\|x^{(u+j-1)}-x^*\right\|^2}\right\}, \tag{9}$$

for all $j \in \{2, \ldots, v\}$ such that $\|x^{(u+j)}-x^*\| > \|x^{(u+j-1)}-x^*\|$, then x' is the unique minimizer of $F_{\mu,\rho}(\cdot;x^*)$ on the discrete path.

Proof: Since $\rho > 0, 0 \le \mu \le \rho/(2K^2L^2) < \rho/L^2$, $x^{(0)} \neq x^*$ and $f(x^{(0)}) \ge f(x^*)$, thus, from Lemma 1, we have $F_{\mu,\rho}(x^{(0)}; x^*) < F_{\mu,\rho}(x^*; x^*) = 0$.

Moreover, due to $\rho > 0, 0 \le \mu \le \rho/(2K^2L^2)$, $f(x^{(i)}) \ge f(x^*)$, for $i = 0, 1, 2, \ldots, u$, and $\|x^{(i-1)} - x^*\| < \|x^{(i)} - x^*\|$, for $i = 1, 2, \ldots, u$, then, from Lemmas 4 and 5, $F_{\mu,\rho}(x^{(i)}; x^*) < F_{\mu,\rho}(x^{(i-1)}; x^*) < 0$, for $i = 1, 2, \ldots, u$.

Since d' satisfies (7), so if $0 < \rho < \rho_0$, then

$$F_{\mu,\rho}(x' + d'; x^*) = f(x^*) - f(x' + d') - \rho \|x' + d' - x^*\|^2 > 0.$$

Therefore, x' minimizes $F_{\mu,\rho}(\cdot; x^*)$ on the discrete path $\{x^{(i)}\}_{i=-1}^{u+1}$ in X between x^* and $x' + d'$.

If $x' + d' = x^{**}$, then the theorem is proved. On the other hand, suppose that $x' + d' \neq x^{**}$. Since $x' + d' \in B^{**}$, there exists a discrete steepest descent trajectory $\{x^{(i)}\}_{i=u+1}^{u+v}$ of f in X from $x' + d'$ to x^{**}. Hence, the first part of the theorem is proved.

To prove the remaining part of the theorem for the case that $x' + d' \neq x^{**}$, we consider the following two cases with the fact that $f(x^*) > f(x^{(u+1)}) > f(x^{(u+2)}) > \cdots > f(x^{(u+v)})$:

1. If $\|x^{(u+j-1)} - x^*\| \ge \|x^{(u+j)} - x^*\|$, for some $j \in \{2, \ldots, v\}$, the following holds for any $\rho > 0$:

$$\begin{aligned} F_{\mu,\rho}\left(x^{(u+j-1)}; x^*\right) &= f(x^*) - f\left(x^{(u+j-1)}\right) - \rho\left\|x^{(u+j-1)} - x^*\right\|^2 \\ &< f(x^*) - f\left(x^{(u+j)}\right) - \rho\left\|x^{(u+j)} - x^*\right\|^2 \\ &= F_{\mu,\rho}\left(x^{(u+j)}; x^*\right). \end{aligned}$$

2. On the other hand, if $\|x^{(u+j-1)} - x^*\| < \|x^{(u+j)} - x^*\|$, for some $j \in \{2, \ldots, v\}$, then $F_{\mu,\rho}(x^{(u+j-1)}; x^*) < F_{\mu,\rho}(x^{(u+j)}; x^*)$ when

$$0 < \rho < \frac{f\left(x^{(u+j-1)}\right) - f\left(x^{(u+j)}\right)}{\left\|x^{(u+j)} - x^*\right\|^2 - \left\|x^{(u+j-1)} - x^*\right\|^2}.$$

Therefore, if ρ satisfies (9), then x' is the unique minimizer of $F_{\mu,\rho}(\cdot; x^*)$ on the discrete path. □

From Theorem 8, we conclude that $F_{\mu,\rho}(\cdot; x^*)$ satisfies the discrete filled function condition (D3) if $\rho > 0$ is sufficiently small and $0 \le \mu \le \rho/(2K^2L^2)$.

From Sections 3.1–3.3, we conclude that $F_{\mu,\rho}(\cdot; x^*)$ is a discrete filled function if $\rho > 0$ is sufficient small and $0 \le \mu \le \rho/(2K^2L^2)$.

3.4. Further properties of the proposed discrete filled function

Lemma 9. *For every $x', x^* \in X$, if there exists $i \in \{1, 2, \ldots, n\}$ such that $x' \pm e_i \in X$, then there exists $d \in \{\pm e_i\}$ such that $\|x' + d - x^*\| > \|x' - x^*\|$.*

Proof: If there is an $i \in \{1, 2, \ldots, n\}$ such that $[x']_i \geq [x^*]_i$, where $[x]_i$ is the ith component of any vector $x \in X$, then $d = e_i$. On the other hand, if there is an $i \in \{1, 2, \ldots, n\}$ such that $[x']_i \leq [x^*]_i$, then $d = -e_i$. □

Theorem 10. *Suppose that $x^* \in X$ and $x' \in \text{int } X$ are local minimizers of f and $F_{\mu,\rho}(\cdot; x^*)$ over X, respectively. If $\rho > 0$ and $0 \leq \mu \leq \rho/(2K^2L^2)$, then*

1. *when $f(x') \geq f(x^*)$, then $f(x' + d') < f(x^*)$ for all $d' \in D_1$, where $D_1 \equiv \{d \in D: x' + d \in X,\ \|x' + d - x^*\| > \|x' - x^*\|\}$;*
2. *x' is in a discrete basin of f that is lower than the discrete basin B^* of f at x^*.*

Proof: Suppose that $\rho > 0$, $0 \leq \mu \leq \rho/(2K^2L^2)$ and $f(x') \geq f(x^*)$. Since, from Corollary 2, x^* is a strict local maximizer of $F_{\mu,\rho}(\cdot; x^*)$, but it is assumed that x' is a local minimizer of $F_{\mu,\rho}(\cdot; x^*)$, therefore $x^* \neq x'$. Since $x' \in \text{int } X$, then by Lemma 9, there exists $d' \in D$ such that $x' + d' \in X$ and $\|x' + d' - x^*\| > \|x' - x^*\|$ and thus $D_1 \neq \emptyset$. For every $d' \in D_1$, if $f(x' + d') \geq f(x^*)$, then, by Theorem 6, $F_{\mu,\rho}(x' + d'; x^*) < F_{\mu,\rho}(x'; x^*)$. It contradicts the assumption that x' is a local minimizer of $F_{\mu,\rho}(\cdot; x^*)$. Hence, $f(x' + d') < f(x^*)$ for all $d' \in D_1$. Therefore, the first part of the theorem holds.

Moreover, it is obvious that any point $x' \in \{x \in X : f(x) < f(x^*)\}$ is in a lower discrete basin of f than B^*. If $f(x') \geq f(x^*)$, the first part of the theorem shows that $f(x' + d') < f(x^*)$, for all $d' \in D_1$. Now, let d^* be the discrete steepest descent direction at x' of f over X. Thus, $f(x' + d^*) \leq f(x' + d') < f(x^*)$. Moreover, x' and $x' + d^*$ are belonging to the same discrete basin. Therefore, x' is in a discrete basin of f that is lower than B^*. □

Theorem 10 shows a very nice property of the discrete filled function proposed in this paper: Every local minimizer of the discrete filled function is in a discrete basin of f that is lower than the current discrete basin of f.

4. Numerical implementation and solution algorithm

Based on the theoretical results in the previous section, a global optimization algorithm (Algorithm 2) is proposed in this section. In the following, we first present the algorithm. We then explain the motivation and mechanism behind the algorithm. Through the algorithm, we denote by x the current iterative point. At the kth iteration of the algorithm, we denote by $x_k^{(0)}$ the feasible initial point; x_k' the local minimizer of $F_{\mu,\rho}(\cdot; x^*)$ on a discrete path; and x_k^* the local minimizer of f over X obtained by the algorithm.

Algorithm 2 (Discrete filled function method)

1. *Input*:
   (a) *an initial point $x_0^{(0)} \in X$;*
   (b) *two fractions: $\hat{\rho}$ $(0 < \hat{\rho} < 1)$ and $\hat{\mu}$ $(0 < \hat{\mu} < 1)$;*
   (c) *the lower bound of ρ, namely $\rho_L > 0$.*

2. *Initialization*:
   (a) *starting from the user provided initial point* $x_0^{(0)}$, *minimize* f *and obtain an initial local minimizer* x_0^* *of* f *over* X;
   (b) *set the iteration count to zero, i.e.,* $k := 0$;
   (c) *set* $\rho, \mu := 1$: *the initial parameters of the proposed discrete filled function in* (3).
3. *Generate initial points*:
   (a) *generate a set of* m *initial points for the proposed discrete filled function method, namely* $\{x_k^{(0)i} \in X \setminus x^*: i = 1, 2, \ldots, m\}$;
   (b) *initialize the vector of the trial order of the initial points, namely set* $J := [1, 2, \ldots, m]$ *and* $j := 1$.
4. *Set* $i := J_j$ *and* $x := x_k^{(0)i}$.
5. *If* $f(x) < f(x_k^*)$, *then*
   (a) *use* x *as an initial point for a discrete local minimization method to find* x_{k+1}^* *such that* $f(x_{k+1}^*) < f(x_k^*)$;
   (b) *increase the iteration count by one, i.e., set* $k := k + 1$;
   (c) *go to Step* 3.
6. *Let* $D_0 := \{d \in D: x + d \in X\}$. *If there exists* $d \in D_0$ *such that* $f(x + d) < f(x_k^*)$, *then*
   (a) *use* $x + d^*$, *where* $d^* = \arg\min_{d \in D_0}\{f(x + d)\}$, *as an initial point for a discrete local minimization method to find* x_{k+1}^* *such that* $f(x_{k+1}^*) < f(x_k^*)$;
   (b) *increase the iteration count by one, i.e., set* $k := k + 1$;
   (c) *go to Step* 3.
7. *Let* $D_1 := \{d \in D_0: \|x + d - x^*\| > \|x - x^*\|\}$. *If* $D_1 = \emptyset$, *then go to Step* 10.
8. *If there exists* $d \in D_1$ *such that* $F_{\mu,\rho}(x + d; x^*) \geq F_{\mu,\rho}(x; x^*)$, *then*
   (a) *reduce* μ *by a user pre-defined fraction, i.e., set* $\mu := \hat{\mu}\mu$;
   (b) *set* $J := [J_j, \ldots, J_m, J_1, \ldots, J_{j-1}]$;
   (c) *set* $j := 1$;
   (d) *go to Step* 4.
9. *Let* $D_2 := \{d \in D_1: f(x + d) < f(x)\}$. *If* $D_2 \neq \emptyset$, *then set*

$$d^* := \arg\min_{d \in D_2}\{f(x + d) + F_{\mu,\rho}(x + d; x^*)\}$$

   otherwise set

$$d^* := \arg\min_{d \in D_1}\{F_{\mu,\rho}(x + d; x^*)\}.$$

   After that, set $x := x + d^*$ *and go to Step* 6.
10. *If* $j < m$, *then set* $j := j + 1$ *and go to Step* 4.

11. *Reduce ρ by a user pre-defined fraction, i.e., set $\rho := \hat{\rho}\rho$. If $\rho \geq \rho_L$, then go to Step* 3. *On the other hand, if $\rho < \rho_L$, then the algorithm is incapable of finding a better local minimizer starting from the initial points, $\{x_k^{(0)i}: i = 1, 2, \ldots, m\}$. The algorithm stops and $x^*_{\text{global}} := x_k^*$ is taken as a global minimizer.*

The motivation and mechanism behind the algorithm are explained below.

A set of m initial points is generated in Step 3 to minimize the discrete filled function. According to Corollary 2, the current local minimizer x_k^* of f over X is a strict maximizer of $F_{\mu,\rho}(\cdot; x^*)$. If no additional information about the objective function is provided, we set the initial points symmetric about x_k^*. For example, we would set $m = 2n$ and choose $x_k^* \pm e_i$, for $i = 1, 2, \ldots, n$, as initial points for the discrete filled function method.

Step 5 represents the situation where the current computer generated initial point for the discrete filled function method satisfies $f(x) < f(x_k^*)$. Therefore, we can further minimize the primal objective function f by any discrete local minimization method starting from x. Note that, Step 5 is needed only if we choose some initial points outside the associated discrete basin B_k^* of f at x_k^* over X.

Step 6 is based on Theorems 8 and 10. If the current iterative point x is in the discrete basin B_{k+1}^* of f at x_{k+1}^* over X and there exists a feasible direction such that $x + d \in X$ and $f(x + d) < f(x_k^*)$, then we switch from minimization of the discrete filled function $F_{\mu,\rho}(\cdot; x^*)$ to a local search for the primal objection function f using $x + d^*$ as an initial point, where d^* is the discrete steepest descent direction of f at x over X.

Steps 7 and 8 are due to Theorem 6. First, we define a set D_1 as all feasible directions that can move further away from x_k^* than x. If D_1 is an empty set, we restart the searching procedure using another initial point. On the other hand, if D_1 is a nonempty set, then Theorem 6 guarantees that every $d \in D_1$ holds either $f(x + d) < f(x_k^*)$ or $F_{\mu,\rho}(x + d; x^*) < F_{\mu,\rho}(x; x^*)$ if μ is chosen small enough. Notice that if the algorithm goes from Steps 6 to 7, every $d \in D_1$ must satisfy $f(x + d) \geq f(x_k^*)$. Thus, if there exists $d \in D_1$ such that $F_{\mu,\rho}(x + d; x^*) \geq F_{\mu,\rho}(x; x^*)$, it implies that μ is not chosen small enough. Therefore, the value of μ is reduced by a fraction and the searching procedure is restarted. Since the initial points $x^{(0)i}$, for $i = J_1, \ldots, J_{j-1}$, did not lead to a better local minimizer with the previous μ, they are not as promising as the remaining initial points, we reorder the trial-order list in the vector J.

Step 9 aims at selecting a more promising successor point. Note that if the algorithm goes from Steps 8 to 9, every $d \in D_1$ must satisfy $f(x + d) \geq f(x_k^*)$ and $F_{\mu,\rho}(x + d; x^*) < F_{\mu,\rho}(x; x^*)$. If there exists $d \in D_1$ such that $f(x + d) < f(x)$, we reduce both the objective function and the discrete filled function at the same time in order to take advantages of their reductions, as suggested in [30]. On the other hand, if every $d \in D_1$ is an increasing direction of f at x, we reduce the discrete filled function alone.

Recall from Theorem 8 that the value of ρ should be selected small enough. Otherwise, there could be no minimizer of $F_{\mu,\rho}(\cdot; x^*)$ in a better basin. Thus, the value of ρ is reduced successively in the solution process in Step 11 if no better solution is found when minimizing the discrete filled function. If the value of ρ reaches its lower bound ρ_L and no better solution is found, Corollary 2 implies that the current local minimizer is probably a global minimizer. Therefore, the algorithm stops and x_k^* is taken as a global minimizer.

5. Numerical experiment

In this section, the proposed solution algorithm is programmed in Matlab 5.3 Release 11 for working on the Sun Blade 2000 system with 900 MHz CPU. In the following, we first give an illustrative example to show the solution procedure of the algorithm described in the previous section. We then report the results of the algorithm in solving several test problems. Some of those test problems are taken from the literature and some of them are the discrete counterparts of some continuous or mixed global optimization problems as adopted in [22] and [31].

Through out the tests, we use the discrete steepest descent method (Algorithm 1) to perform local searches. Suppose that we obtain a local minimizer x_k^* of f over X at the kth iteration using $x_{k-1}^* + e_j$ as the initial point, where x_{k-1}^* is the local minimizer of f obtained at the $(k-1)$st iteration, we then use the neighboring points of x_k^* as the initial points in minimizing the discrete filled function in the following order:

$$
\begin{aligned}
&x_k^* + e_j,\\
&x_k^* + e_{j+1}, x_k^* - e_{j+1}, \ldots, x_k^* + e_n, x_k^* - e_n,\\
&x_k^* + e_1, x_k^* - e_1, \ldots, x_k^* + e_{j-1}, x_k^* - e_{j-1},\\
&x_k^* - e_j.
\end{aligned}
$$

Notice that, if $x_k^* \in \partial X$, we would have less than $2n$ initial points. In addition, we set $\rho = \rho_L = 1$ in the tests. In other words, if the algorithm could not find a minimizer of $F_{\mu,\rho}(\cdot; x^*)$ using all initial points, we stop the algorithm immediately. Besides these, we set $\mu = 1$ at the beginning of the algorithm. Once the current μ is classified as insufficiently small, we reduce μ to $\mu/10$.

In each test problem, we will first give a mathematical model. After that, we will summarize the computational results of the proposed method with the user-provided initial point: $x_0^{(0)}$; the global minimizer or the best known solution: x_{global}^*; the number of iteration cycles: #iter; the CPU time in seconds to obtain the final results: $\text{CPU}_{\text{final}}$; the CPU time in seconds for the algorithm to stop at Step 11: CPU_{stop}; the total numbers of objective function evaluations to obtain the final results and to stop at Step 11: # f_{final} and # f_{stop}, respectively.

Now, we consider the following illustrative example.

Example 3 (Example 5.2 in [8], Problem 1 in [29]).

$$
\begin{aligned}
\min \quad & f(x) = x_1 + 10x_2\\
\text{s.t.} \quad & 66x_1 + 14x_2 \geq 1430,\\
& -82x_1 + 28x_2 \geq 1306,\\
& 0 \leq x_1 \leq 15, \quad 68 \leq x_2 \leq 102, \quad x_1, x_2 \text{ integers.}
\end{aligned}
$$

This problem is a LINEAR INTEGER PROGRAMMING PROBLEM. It has 314 feasible points where seven of them are local minimizers and one of these local minimizers is the global minimum solution: $x_{\text{global}}^* = (7, 70)$ with $f(x_{\text{global}}^*) = 707$.

We start from a feasible point $x_0^{(0)} = (15, 102)$ with $f(x_0^{(0)}) = 1035$ and use the discrete steepest descent method to minimize f. After 30 function evaluations, we obtain an initial local minimizer $x_0^* = (3, 88)$ with $f(x_0^*) = 883$.

In the first iteration of the algorithm, $\mu = 1$, 0.1 and 0.01 are found to be not sufficiently small enough. When $\mu = 0.001$, we start from $x_1^{(0)} = (4, 88)$ and reach $x_1' = (4, 87)$ with $f(x_1') = 874 < f(x_0^*)$. We then switch to the local search again and obtain $x_1^* = (4, 84)$ with $f(x_1^*) = 844$. The cumulative number of function evaluations is 38.

In the second iteration of the algorithm, we start from $x_2^{(0)} = (5, 84)$ and reach $x_2' = (5, 83)$ with $f(x_2') = 835 < f(x_1^*)$. We then switch to the local search and obtain $x_2^* = (5, 79)$ with $f(x_2^*) = 795$. The cumulative number of function evaluations is 47.

In the same fashion, we have $x_3^{(0)} = (6, 79)$, $x_3' = (6, 78)$ with $f(x_3') = 786 < f(x_2^*)$, $x_3^* = (6, 74)$ with $f(x_3^*) = 746$, $x_4^{(0)} = (7, 74)$, $x_4' = (7, 73)$ with $f(x_4') = 737 < f(x_3^*)$, $x_4^* = (7, 70)$ with $f(x_4^*) = 707$ and the cumulative numbers of function evaluations are 56 and 63 in the third and fourth iterations, respectively.

In the fifth iteration of the algorithm, three starting points, (8, 70), (6, 70) and (7, 69), are infeasible. Besides these, the algorithm cannot find a feasible point with function value less than $f(x_4^*)$ using the remaining starting point (7, 71). The cumulative number of function evaluations is 94.

In general, we should reduce ρ by a fraction and continue the process until $\rho < \rho_L$. Since $\rho = \rho_L = 1$ is selected in the numerical tests, and thus the algorithm is terminated. Therefore, we take # iter = 4, $x_{\text{global}}^* = x_4^* = (7, 70)$, $f(x_{\text{global}}^*) = f(x_4^*) = 707$, # $f_{\text{final}} =$ 63 and # $f_{\text{stop}} = 94$. The ratio of the number of function evaluations to reach the global minimum to the number of feasible points is $63/314 \approx 0.2006$.

In the following, we report some numerical results of the algorithm in solving several test problems.

Problem 1 (Example 4.3 in [2], Problem 8 in [22])**.**

$$
\begin{aligned}
\min \quad & f(x) = x_1^2 + x_2^2 + 3x_3^2 + 4x_4^2 + 2x_5^2 - 8x_1 - 2x_2 - 3x_3 - x_4 - 2x_5, \\
\text{s.t.} \quad & x_1 + 2x_2 + 2x_3 + x_4 + 6x_5 \le 800, \\
& 2x_1 + x_2 + 6x_3 \le 200, \\
& x_3 + x_4 + 5x_5 \le 200, \\
& x_1 + x_2 + x_3 + x_4 \ge 48, \\
& x_2 + x_4 + x_5 \ge 34, \\
& 6x_1 + 7x_5 \ge 104, \\
& 55 \le x_1 + x_2 + x_3 + x_4 + x_5 \le 400, \\
& 0 \le x_i \le 99, \quad x_i \text{ integer}, \quad i = 1, 2, 3, 4, 5.
\end{aligned}
$$

This problem is a QUADRATIC INTEGER PROGRAMMING PROBLEM. It has 251401581 feasible points. The solution to the problem was given in [2] as $x_{\text{global}}^* = (17, 18, 7, 7, 9)$ with $f(x_{\text{global}}^*) = 900$. We used five initial points in our experiment: $x_0^{(0)} = (17, 18, 7, 7, 9)$,

(21, 34, 0, 0, 0), (0, 0, 0, 48, 15), (100, 0, 0, 0, 40) and (0, 8, 32, 8, 32). For every experiment, the proposed method succeeded in identifying a better minimum solution $x^*_{\text{global}} = (16, 22, 5, 5, 7)$ with $f(x^*_{\text{global}}) = 807$, which is the same as the optimal solution given in [22]. Moreover, the maximum numbers of function evaluations to reach the global minimum and to stop were only 6876 and 10827, respectively. They were much smaller than the average number of function evaluations (187794) reported in [22]. The maximum CPU time to reach the global minimum was about 11.57 seconds. The ratio of the maximum number of function evaluations to reach the global minimum to the number of feasible points was about 2.74×10^{-5}. A summary of the computational results is displayed in Table 1 in the Appendix.

Problem 2 (Example 4.1 in [2], Problem 2 in [22], Problem 3 in [29])**.**

$$\begin{aligned} \max \quad & f(x) = x_1^2 + x_1x_2 - x_2^2 + x_3x_1 - x_3^2 + 8x_4^2 - 17x_5^2 + 6x_6^3 \\ & \qquad + x_6x_5x_4x_7 + x_8^3 + x_9^4 - x_{10}^5 - x_{10}x_5 + 18x_3x_7x_6, \\ \text{s.t.} \quad & 0 \le x_i \le 99, \quad x_i \text{ integer}, \quad i = 1, 2, \ldots, 10. \end{aligned}$$

This problem is a BOX CONSTRAINED/UNCONSTRAINED NONLINEAR INTEGER PROGRAMMING PROBLEM. It has 10^{20} feasible points. The objective function value of the global solution to the problem as given in [2] is 197580315. Mohan and Nguyen [22] reported a better solution with function value 209205232. We used ten initial points in our experiment: $x_0^{(0)} = (0, \ldots, 0)$, $(20, \ldots, 20)$, $(40, \ldots, 40)$, $(60, \ldots, 60)$, $(80, \ldots, 80)$, $(99, \ldots, 99)$, $(0, \ldots, 0, 0, 99, \ldots, 99)$, $(99, \ldots, 99, 99, 0, \ldots, 0)$, $(0, 99, 0, 99, \ldots)$ and $(99, 0, 99, 0, \ldots)$. For every experiment, the proposed method succeeded in identifying a better solution: $x^*_{\text{global}} = (99, 49, 99, 99, 99, 99, 99, 99, 99, 0)$ with $f(x^*_{\text{global}}) = 216300719$, during initialization (Step 2). Moreover, the maximum number of function evaluations to reach the new best known solution was only 942. It was much smaller than the average number of function evaluations (2695) reported in [22]. The maximum CPU time to reach the new best known solution was about 0.36 seconds. The ratio of the maximum number of function evaluations to reach the new best known solution to the number of feasible points was about 9.42×10^{-18}. A summary of the computational results is displayed in Table 2 in the Appendix.

Problem 3 (Problem 16 in [22])**.**

$$\begin{aligned} \min \quad & f(u, v) = c^T u - \frac{1}{2} u^T Q u + d^T v, \\ \text{s.t.} \quad & 2u_1 + 2u_2 + v_6 + v_7 \le 10, \\ & 2u_1 + 2u_3 + v_6 + v_8 \le 10, \\ & 2u_2 + 2u_3 + v_7 + v_8 \le 10, \\ & -2u_4 - v_1 + v_6 \le 0, \\ & -2v_2 - v_3 + v_7 \le 0, \\ & -2v_4 - v_5 + v_8 \le 0, \\ & -8u_i + v_{i+5} \le 0, \quad i = 1, 2, 3, \end{aligned}$$

$$
\begin{aligned}
& u_j, v_k \in \{0, 1\}, \qquad j = 1, 2, 3, 4, \quad k = 1, 2, 3, 4, 5, 9, \\
& v_l \in \{0, 1, 2, 3\}, \qquad l = 6, 7, 8, \\
\text{where } & c^T = (5, 5, 5, 5), \quad Q = 10 * I_4, \; d^T = (-1, \ldots, -1), \quad x = (u, v).
\end{aligned}
$$

This problem is a QUADRATIC INTEGER PROGRAMMING PROBLEM. It has 5488 feasible points where 1923 of them are local minimizers but only sixteen of those local minimizers are the global minimum solutions: $x^*_{\text{global}} = (u_1, u_2, u_3, u_4, 1, 1, 1, 1, 1, 3, 3, 3, 1)$, for all $u_1, u_2, u_3, u_4 \in \{0, 1\}$, with $f(x^*_{\text{global}}) = -15$. We used ten randomly generated initial points in our experiment. For every experiment, the proposed method succeeded in identifying a global minimum solution. Moreover, the maximum numbers of function evaluations to reach the global minimum and to stop were only 180 and 868, respectively. They were much smaller than the average number of function evaluations (7887) reported in [22]. The maximum CPU time to reach the global minimum was about 0.26 seconds. The ratio of the maximum number of function evaluations to reach the global minimum to the number of feasible points was about 3.28×10^{-2}. A summary of the computational results is displayed in Table 3 in the Appendix.

Problem 4.

$$
\begin{aligned}
\min \quad & f(x) = \sum_{i=1}^{9} \left\{ \exp\left[-\frac{(u_i - x_2)^{x_3}}{x_1} \right] - \frac{i}{100} \right\}^2, \\
\text{s.t.} \quad & 1 \le x_1 \le 100, \quad 0 \le x_2 \le 25, \quad x_i \text{ integer}, \quad i = 1, 2, \\
& x_3 = j/2, \quad 0 \le j \le 10, \quad j \text{ integer}, \\
\text{where} \quad & u_i = 25 + [-50 \log(i/100)]^{2/3}.
\end{aligned}
$$

This problem is a discrete counterpart of the problem 1 in [22]. It is a BOX CONSTRAINED/UNCONSTRAINED NONLINEAR INTEGER PROGRAMMING PROBLEM. It has 28600 feasible points and many local minimizers. More precisely, it has 12555 and 1484 local minimizers in the interior and on the boundary of the box constraints, respectively. Nevertheless, it has only one global minimum solution: $x^*_{\text{global}} = (50, 25, 1.5)$ with $f(x^*_{\text{global}}) \approx 0$. We used nine initial points in our experiment: $x_0^{(0)} = (1, 0, 0)$, $(25, 6, 1)$, $(50, 12, 2.5)$, $(75, 18, 3.5)$, $(100, 25, 5)$, $(1, 0, 5)$, $(100, 25, 0)$, $(1, 25, 0)$ and $(100, 0, 5)$. For every experiment, the proposed method succeeded in identifying the global minimum solution. The maximum CPU time to reach the global minimum was about 4.10 seconds. The ratio of the maximum number of function evaluations to reach the global minimum to the number of feasible points was about 0.1117. A summary of the computational results is displayed in Table 4 in the Appendix.

Problem 5 (Colville's function 4)**.**

$$
\begin{aligned}
\min \quad & f(x) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2 + 90\left(x_4 - x_3^2\right)^2 + (1 - x_3)^2 \\
& \qquad + 10.1[(x_2 - 1)^2 + (x_4 - 1)^2] + 19.8(x_2 - 1)(x_4 - 1), \\
\text{s.t.} \quad & -10 \le x_i \le 10, \quad x_i \text{ integer}, \quad i = 1, 2, 3, 4.
\end{aligned}
$$

This problem is a discrete counterpart of the problem 38 in [14] and the problem 260 in [25]. It is a BOX CONSTRAINED/UNCONSTRAINED NONLINEAR INTEGER PROGRAMMING PROBLEM. It has $21^4 \approx 1.94 \times 10^5$ feasible points where 41 of them are local minimizers but only one of those local minimizers is the global minimum solution: $x^*_{\text{global}} = (1, 1, 1, 1)$ with $f(x^*_{\text{global}}) = 0$. We used nine initial points in our experiment: $x_0^{(0)} = (-10, \ldots, -10)$, $(-5, \ldots, -5), (0, \ldots, 0), (5, \ldots, 5), (10, \ldots, 10), (-10, -10, 10, 10), (10, 10, -10, -10)$, $(-10, 10, -10, 10)$ and $(10, -10, 10, -10)$. For every experiment, the proposed method succeeded in identifying the global minimum solution. The maximum CPU time to reach the global minimum was about 5.08 seconds. The ratio of the maximum number of function evaluations to reach the global minimum to the number of feasible points was about 1.77×10^{-2}. A summary of the computational results is displayed in Table 5 in the Appendix.

Problem 6 (Goldstein and Price's function, Problem D.1 in [31])**.**

$$\begin{aligned} \min \quad & f(x) = g(x)h(x), \\ \text{s.t.} \quad & x_i = 0.001j, \quad -2000 \le j \le 2000, \quad j \text{ integer}, \quad i = 1, 2, \end{aligned}$$

where

$$\begin{aligned} g(x) &= 1 + (x_1 + x_2 + 1)^2\big(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2\big), \\ h(x) &= 30 + (2x_1 - 3x_2)^2\big(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\big). \end{aligned}$$

This problem is a discrete counterpart of the Goldstein and Price's function in [11] and was adopted in [31]. It is a BOX CONSTRAINED/UNCONSTRAINED NONLINEAR INTEGER PROGRAMMING PROBLEM. It has $4001^2 \approx 1.60 \times 10^7$ feasible points and many local minimizers. More precisely, it has 207 and 2 local minimizers in the interior and on the boundary of the box constraints, respectively. Nevertheless, it has only one global minimum solution: $x^*_{\text{global}} = (0, -1)$ with $f(x^*_{\text{global}}) = 3$. We used seven initial points in our experiment: $x_0^{(0)} = (-2, -2), (-1, -1), (0, 0), (1, 1), (2, 2), (-2, 2)$ and $(2, -2)$. For every experiment, the proposed method succeeded in identifying the global minimum solution. The maximum CPU time to reach the global minimum was about 50.52 seconds. The ratio of the maximum number of function evaluations to reach the global minimum to the number of feasible points was about 2.53×10^{-3}. A summary of the computational results is displayed in Table 6 in the Appendix.

Problem 7 (Beale's function)**.**

$$\begin{aligned} \min \quad & f(x) = [1.5 - x_1(1 - x_2)]^2 + \big[2.25 - x_1\big(1 - x_2^2\big)\big]^2 \\ & \qquad + \big[2.625 - x_1\big(1 - x_2^3\big)\big]^2, \\ \text{s.t.} \quad & x_i = 0.001j, \quad -10^4 \le j \le 10^4, \quad j \text{ integer}, \quad i = 1, 2. \end{aligned}$$

This problem is a discrete counterpart of the problem 5 in [23] and the problem 203 in [25]. It is a BOX CONSTRAINED/UNCONSTRAINED NONLINEAR INTEGER PROGRAMMING PROBLEM. It has $20001^2 \approx 4.00 \times 10^8$ feasible points and many local minimizers, but only one global minimum solution: $x^*_{\text{global}} = (3, 0.5)$ with $f(x^*_{\text{global}}) = 0$. We used seven initial points in our experiment: $x_0^{(0)} = (-10, -10), (-5, -5), (0, 0), (5, 5), (10, 10), (-10, 10)$ and $(10, -10)$. For every experiment, the proposed method succeeded in identifying the global minimum solution. The maximum CPU time to reach the global minimum was about 19.39 minutes. The ratio of the maximum number of function evaluations to reach the global minimum to the number of feasible points was about 2.61×10^{-3}. A summary of the computational results is displayed in Table 7 in the Appendix.

Problem 8.

$$\begin{aligned} \min \quad & f(x) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2, \\ \text{s.t.} \quad & x_1^2 + x_2^2 \geq 0.25, \quad -\frac{1}{3}x_1 + x_2 \geq -0.1, \\ & x_i = j_i \times 10^{-4}, \quad 0 \leq j_i \leq 10^5, \quad j_i \text{ integer}, \quad i = 1, 2. \end{aligned}$$

This problem is a combination of the discrete counterparts of the problems 231 and 233 in [25]. It is a NONCONVEX NONLINEAR INTEGER PROGRAMMING PROBLEM. It has about 1.00×10^{10} points in the box $0 \leq x_i \leq 10$ $(i = 1, 2)$, but not all of them are feasible. It has many local minimizers, but only one global minimum solution: $x^*_{\text{global}} = (1, 1)$ with $f(x^*_{\text{global}}) = 0$. We used nine initial points in our experiment: $x_0^{(0)} = (2, 2), (4, 4), (6, 6), (8, 8), (10, 10), (0, 0.5), (0, 10), (10, 3.2334)$ and $(0.3536, 0.3536)$. For every experiment, the proposed method succeeded in identifying the global minimum solution. The maximum CPU time to reach the global minimum was about 52.21 minutes. The ratio of the maximum number of function evaluations to reach the global minimum to the number of points in the box was about 2.47×10^{-4}. A summary of the computational results is displayed in Table 8 in the Appendix.

Problem 9 (Powell's singular function)**.**

$$\begin{aligned} \min \quad & f(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4, \\ \text{s.t.} \quad & x_i = 0.001\, j, \quad -10^4 \leq j \leq 10^4, \quad j \text{ integer}, \quad i = 1, 2, 3, 4. \end{aligned}$$

This problem is a discrete counterpart of the problem 13 in [23] and the problem 256 in [25]. It is a BOX CONSTRAINED/UNCONSTRAINED NONLINEAR INTEGER PROGRAMMING PROBLEM. It has $20001^4 \approx 1.60 \times 10^{17}$ feasible points and many local minimizers, but only one global minimum solution: $x^*_{\text{global}} = (0, 0, 0, 0)$ with $f(x^*_{\text{global}}) = 0$. We used ten initial points in our experiment: $x_0^{(0)} = (-10, \ldots, -10), (-6, \ldots, -6), (-3, \ldots, -3), (3, \ldots, 3), (6, \ldots, 6), (10, \ldots, 10), (-10, -10, 10, 10), (10, 10, -10, -10), (-10, 10, -10, 10)$ and $(10, -10, 10, -10)$. For every experiment, the proposed method succeeded in identifying the global minimum solution. The maximum CPU time to reach the global minimum was

about 1.71 hours. The ratio of the maximum number of function evaluations to reach the global minimum to the number of feasible points was about 3.45×10^{-11}. A summary of the computational results is displayed in Table 9 in the Appendix.

Problem 10.

$$\begin{aligned} \min \quad & f(x) = (x_1 - 1)^2 + (x_n - 1)^2 + n \sum_{i=1}^{n-1} (n - i)\left(x_i^2 - x_{i+1}\right)^2, \\ \text{s.t.} \quad & -5 \le x_i \le 5, \quad x_i \text{ integer}, \quad i = 1, 2, \ldots, n. \end{aligned}$$

This problem is a generalization of the problem 282 in [25]. It is a BOX CONSTRAINED/UNCONSTRAINED NONLINEAR INTEGER PROGRAMMING PROBLEM. It has 11^n feasible points and many local minimizers (4, 6, 7, 10 and 12 minimizers for $n = 2$, 3, 4, 5 and 6, respectively), but only one global minimum solution: $x^*_{\text{global}} = (1, \ldots, 1)$ with $f(x^*_{\text{global}}) = 0$, for all n. We considered three sizes of the problem: $n = 25$, 50 and 100. In other words, there were about 1.08×10^{26}, 1.17×10^{52} and 1.38×10^{104} feasible points, for $n = 25$, 50 and 100, respectively. For all problems with different sizes, we used nine initial points in our experiment: $x_0^{(0)} = (-5, \ldots, -5), (-3, \ldots, -3), (0, \ldots, 0), (3, \ldots, 3), (5, \ldots, 5), (-5, \ldots, -5, -5, 5, \ldots, 5), (5, \ldots, 5, 5, -5, \ldots, -5), (-5, 5, -5, 5, \ldots)$ and $(5, -5, 5, -5, \ldots)$. For every experiment, the proposed method succeeded in identifying the global minimum solution. The maximum CPU times to reach the global minima were about 5.54 minutes, 11.82 seconds and 5.63 hours, for $n = 25$, 50 and 100, respectively. The ratios of the maximum numbers of function evaluations to reach the global minima to the numbers of feasible points were about 2.51×10^{-21}, 1.59×10^{-48} and 1.14×10^{-97}, for $n = 25$, 50 and 100, respectively. A summary of the computational results is displayed in Table 10 in the Appendix.

Problem 11 (Rosenbrock's function)**.**

$$\begin{aligned} \min \quad & f(x) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2\right], \\ \text{s.t.} \quad & -5 \le x_i \le 5, \quad x_i \text{ integer}, \quad i = 1, 2, \ldots, n. \end{aligned}$$

This problem is a generalization of the problems 294–299 in [25]. It is a BOX CONSTRAINED/UNCONSTRAINED NONLINEAR INTEGER PROGRAMMING PROBLEM. It has 11^n feasible points and many local minimizers (5, 6, 7, 9 and 11 minimizers for $n = 2$, 3, 4, 5 and 6, respectively), but only one global minimum solution: $x^*_{\text{global}} = (1, \ldots, 1)$ with $f(x^*_{\text{global}}) = 0$, for all n. We considered three sizes of the problem: $n = 25$, 50 and 100. In other words, there were about 1.08×10^{26}, 1.17×10^{52} and 1.38×10^{104} feasible points, for $n = 25$, 50 and 100, respectively. For all problems with different sizes, we used nine initial points in our experiment: $x_0^{(0)} = (-5, \ldots, -5), (-3, \ldots, -3), (0, \ldots, 0), (3, \ldots, 3), (5, \ldots, 5), (-5, \ldots, -5, -5, 5, \ldots, 5), (5, \ldots, 5, 5, -5, \ldots, -5), (-5, 5, -5, 5, \ldots)$ and $(5, -5, 5, -5, \ldots)$. For every experiment, the proposed method succeeded in identifying

the global minimum solution. The maximum CPU times to reach the global minima were about 4.37 minutes, 35.27 minutes and 4.76 hours, for $n = 25$, 50 and 100, respectively. The ratios of the maximum numbers of function evaluations to reach the global minima to the numbers of feasible points were about 1.91×10^{-21}, 1.40×10^{-46} and 9.58×10^{-98}, for $n = 25$, 50 and 100, respectively. A summary of the computational results is displayed in Table 11 in the Appendix.

Problem 12.

$$\begin{aligned} \min \quad & f(x) = \sum_{i=1}^{n} x_i^4 + \left(\sum_{i=1}^{n} x_i \right)^2 \\ \text{s.t.} \quad & -5 \le x_i \le 5, \quad x_i \text{ integer}, \quad i = 1, 2, \ldots, n. \end{aligned}$$

This problem is a BOX CONSTRAINED/UNCONSTRAINED NONLINEAR INTEGER PROGRAMMING PROBLEM. It has 11^n feasible points and many local minimizers (3, 7, 19, 51 and 141 minimizers for $n = 2$, 3, 4, 5 and 6, respectively), but only one global minimum solution: $x^*_{\text{global}} = (0, \ldots, 0)$ with $f(x^*_{\text{global}}) = 0$, for all n. We considered three sizes of the problem: $n = 25, 50$ and 100. In other words, there were about 1.08×10^{26}, 1.17×10^{52} and 1.38×10^{104} feasible points, for $n = 25, 50$ and 100, respectively. For all problems with different sizes, we used ten initial points in our experiment: $x_0^{(0)} = (-5, \ldots, -5), (-3, \ldots, -3), (-1, \ldots, -1)$ $(1, \ldots, 1)$, $(3, \ldots, 3)$, $(5, \ldots, 5)$, $(-5, \ldots, -5, -5, 5, \ldots, 5)$, $(5, \ldots, 5, 5, -5, \ldots, -5)$, $(-5, 5, -5, 5, \ldots)$ and $(5, -5, 5, -5, \ldots)$. For every experiment, the proposed method succeeded in identifying the global minimum solution. The maximum CPU times to reach the global minima were about 3.96 minutes, 31.81 minutes, and 4.34 hours, for $n = 25$, 50 and 100, respectively. The ratios of the maximum numbers of function evaluations to reach the global minima to the numbers of feasible points were about 1.93×10^{-21}, 1.39×10^{-46} and 9.42×10^{-98}, for $n = 25$, 50 and 100, respectively. A summary of the computational results is displayed in Table 12 in the Appendix.

Problem 13.

$$\begin{aligned} \min \quad & f(x) = x^T Q x, \\ \text{s.t.} \quad & \sum_{i=1}^{n} \frac{x_i^2}{9n + i} \le 1, \\ & \sum_{i=1}^{n} i x_i \ge \frac{n}{2}, \\ & -5 \le x_i \le 5, \quad x_i \text{ integer}, \quad i = 1, 2, \ldots, n, \\ \text{where} \quad & Q = [Q_{ij}], \quad Q_{ii} = 2, \quad Q_{ij} = 1, \quad \text{for } i \ne j. \end{aligned}$$

This problem is a CONVEX NONLINEAR INTEGER PROGRAMMING PROBLEM. It has 11^n points in the box $-5 \leq x_i \leq 5$ $(i = 1, 2, \ldots, n)$ but not all of them are feasible. It has many local minimizers, but only a few of them are global minimizers. More precisely, it has $\lfloor n/2 \rfloor (\lfloor n/2 \rfloor + 1)/2$ global minimum solutions: $x^*_{\text{global}} = (0, \ldots, 0, -1, 0, \ldots, 0, 1, 0, \ldots, 0)$ with $f(x^*_{\text{global}}) = 2$, where $[x^*_{\text{global}}]_i = -1$, $[x^*_{\text{global}}]_j = 1$, $j \geq \lceil n/2 \rceil + 1$, $i \leq j - \lceil n/2 \rceil$, for all n. We considered four sizes of the problem: $n = 25$, 50, 100 and 200. In other words, there were about 1.08×10^{26}, 1.17×10^{52}, 1.38×10^{104} and 1.90×10^{208} points in the box $-5 \leq x_i \leq 5$ $(i = 1, 2, \ldots, n)$, for n = 25, 50, 100 and 200, respectively. For all problems with different sizes, we used six initial points in our experiment: $x_0^{(0)} = (\alpha, \ldots, \alpha)$ and $(\alpha, -\alpha, \alpha, -\alpha, \ldots)$, for $\alpha = 1, 2, 3$, when n is odd; $x_0^{(0)} = (\alpha, \ldots, \alpha)$ and $(-\alpha, \alpha, -\alpha, \alpha, \ldots)$, for $\alpha = 1, 2, 3$, when n is even. For every experiment, the proposed method succeeded in identifying a global minimum solution. The maximum CPU times to reach the global minima were about 1.68 minutes, 11.24 minutes, 1.67 hours and 17.44 hours, for $n = 25$, 50, 100 and 200, respectively. The ratios of the maximum numbers of function evaluations to reach the global minima to the numbers of points in the box $-5 \leq x_i \leq 5$ $(i = 1, 2, \ldots, n)$ were about 6.91×10^{-22}, 4.18×10^{-47}, 3.04×10^{-98} and 1.84×10^{201}, for $n = 25$, 50, 100 and 200, respectively. A summary of the computational results is displayed in Table 13 in the Appendix.

6. Conclusions

How to globally solve discrete optimization problems has posed a great challenge in front of the optimization research community. A discrete filled function has been developed in this paper which enables us to move from the current local minimum to a better one, if it exists. Promising computation results have been observed from our numerical experiments of large scales. Recently, global optimality conditions have been derived in [1] for quadratic binary integer programming problems. An integration with global optimality conditions will provide solid stopping conditions for a discrete filled function method.

Appendix

Table 1. Summary of the computational results for Problem 1.

	# iter	$\text{CPU}_{\text{final}}$	CPU_{stop}	# f_{final}	# f_{stop}
$n = 5$		Number of experiments = 5			
Minimum	6	3.91	9.63	2272	6223
Mean	28.20	8.35	14.09	4474.60	8425.60
Median	36.00	8.84	14.61	4407.00	8358.00
Maximum	43	11.57	17.37	6876	10827

Table 2. Summary of the computational results for Problem 2.

	# iter	$\text{CPU}_{\text{final}}$	CPU_{stop}	# f_{final}	# f_{stop}
$n = 10$		Number of experiments = 10			
Minimum	0	0.11	173.11	176	121847
Mean	0.00	0.25	174.64	621.90	122496.90
Median	0.00	0.26	175.19	637.00	122562.00
Maximum	0	0.36	176.51	942	122953

Table 3. Summary of the computational results for Problem 3.

	# iter	$\text{CPU}_{\text{final}}$	CPU_{stop}	# f_{final}	# f_{stop}
$n = 13$		Number of experiments = 10			
Minimum	0	0.07	1.70	50	738
Mean	2.10	0.17	1.76	104.90	792.90
Median	2.00	0.19	1.76	107.00	795.00
Maximum	4	0.26	1.85	180	868

Table 4. Summary of the computational results for Problem 4.

	# iter	$\text{CPU}_{\text{final}}$	CPU_{stop}	# f_{final}	# f_{stop}
$n = 3$		Number of experiments = 9			
Minimum	0	0.04	3.17	29	2532
Mean	10.33	2.01	4.90	1574.22	3904.33
Median	9.00	1.33	4.18	1063.00	3479.00
Maximum	19	4.10	6.71	3194	5305

Table 5. Summary of the computational results for Problem 5.

	# iter	$\text{CPU}_{\text{final}}$	CPU_{stop}	# f_{final}	# f_{stop}
$n = 4$		Number of experiments = 9			
Minimum	1	1.79	4.25	1262	3404
Mean	3.33	2.86	5.51	2121.11	4263.11
Median	4.00	2.34	5.08	1659.00	3801.00
Maximum	6	5.08	7.93	3442	5584

Table 6. Summary of the computational results for Problem 6.

	# iter	CPU_{final}	CPU_{stop}	# f_{final}	# f_{stop}
$n = 2$		Number of experiments = 7			
Minimum	0	0.38	76.23	1007	64533
Mean	16.86	15.46	132.88	13709.86	111125.86
Median	1.00	3.73	151.65	4865.00	126734.00
Maximum	112	50.52	156.86	40498	132975

Table 7. Summary of the computational results for Problem 7.

	# iter	CPU_{final}	CPU_{stop}	# f_{final}	# f_{stop}
$n = 2$		Number of experiments = 7			
Minimum	3	1.51	744.97	3671	632329
Mean	5.14	682.41	1080.43	607880.71	939209.57
Median	4.00	486.12	831.12	421984.00	703758.00
Maximum	11	1163.32	1502.13	1043639	1325413

Table 8. Summary of the computational results for Problem 8.

	# iter	CPU_{final}	CPU_{stop}	# f_{final}	# f_{stop}
$n = 2$		Number of experiments = 9			
Minimum	192	12.09	4081.22	18867	3062103
Mean	202.67	2074.91	4720.13	1608067.33	3562771.33
Median	208.00	3092.60	4995.15	2342322.00	3752760.00
Maximum	208	3132.64	5065.76	2470191	3880629

Table 9. Summary of the computational results for Problem 9.

	# iter	CPU_{final}	CPU_{stop}	# f_{final}	# f_{stop}
$n = 4$		Number of experiments = 10			
Minimum	53	5213.99	7687.18	4692269	6924077
Mean	53	5670.63	8171.75	5105399.50	7337207.50
Median	53	5673.52	8182.02	5102831.50	7334639.50
Maximum	53	6167.48	8717.02	5513394	7745202

Table 10. Summary of the computational results for Problem 10.

	# iter	$\text{CPU}_{\text{final}}$	CPU_{stop}	# f_{final}	# f_{stop}
$n = 25$			Number of experiments = 9		
Minimum	0	0.96	297.95	1451	248497
Mean	1.11	124.65	426.06	102883.22	346738.11
Median	1.00	1.65	318.49	2389.00	253188.00
Maximum	2	332.53	633.95	271472	510362
$n = 50$					
Minimum	0	2.60	2336.51	5401	1970943
Mean	1.11	5.35	2430.30	9840.67	1976994.22
Median	1.00	4.87	2354.87	9477.00	1975114.00
Maximum	2	11.82	2542.74	18634	1990007
$n = 100$					
Minimum	0	9.27	18815.12	20801	15378460
Mean	1.33	8531.79	27942.27	6704633.00	22013196.67
Median	1.00	47.18	20092.05	74759.00	15453726.00
Maximum	4	20252.79	40130.61	15686480	30918516

Table 11. Summary of the computational results for Problem 11.

	# iter	$\text{CPU}_{\text{final}}$	CPU_{stop}	# f_{final}	# f_{stop}
$n = 25$			Number of experiments = 9		
Minimum	0	0.76	292.03	1451	243866
Mean	1.00	110.98	396.72	90586.11	320610.44
Median	1.00	1.77	316.24	2470.00	248607.00
Maximum	2	262.46	535.16	207512	421068
$n = 50$					
Minimum	0	2.76	2150.75	5401	1707270
Mean	1.00	924.07	3093.73	727641.22	2422960.00
Median	1.00	6.16	2196.06	9219.00	1711088.00
Maximum	2	2115.99	4318.59	1641022	3328153
$n = 100$					
Minimum	0	9.91	17245.42	20801	13466632
Mean	1.00	7574.43	25025.92	5861265.44	19293079.56
Median	1.00	23.19	17650.40	35944.00	13481775.00
Maximum	2	17149.97	34788.45	13206805	26621098

Table 12. Summary of the computational results for Problem 12.

	# iter	CPU_{final}	CPU_{stop}	# f_{final}	# f_{stop}
$n = 25$			Number of experiments = 10		
Minimum	11	80.77	367.62	70991	316241
Mean	11.60	131.68	417.85	116765.50	362015.50
Median	12.00	102.00	389.54	90976.50	336226.50
Maximum	12	237.88	526.14	209373	454623
$n = 50$					
Minimum	24	615.15	2919.34	537101	2487001
Mean	24.40	1154.35	3473.98	997241.30	2947141.30
Median	24.00	873.73	3176.16	767103.50	2717003.50
Maximum	25	1908.61	4261.42	1629880	3579780
$n = 100$					
Minimum	48	5115.32	24250.06	4266745	19816545
Mean	49.20	9438.91	28793.14	7770218.80	23320018.80
Median	49.00	6464.21	25941.07	5319711.00	20869511.00
Maximum	50	15608.43	35047.98	12981570	28531370

Table 13. Summary of the computational results for Problem 13.

	# iter	CPU_{final}	CPU_{stop}	# f_{final}	# f_{stop}
$n = 25$			Number of experiments = 6		
Minimum	6	21.40	110.77	15412	76138
Mean	10.00	61.55	153.39	45158.50	107159.17
Median	11.50	61.02	148.67	44490.50	103546.00
Maximum	12	100.97	200.52	74920	141589
$n = 50$					
Minimum	14	177.56	836.97	124046	568296
Mean	20.67	449.51	1152.95	323156.50	798155.17
Median	23.50	493.05	1166.66	353124.50	809832.00
Maximum	24	674.59	1451.81	490831	1014817
$n = 100$					
Minimum	30	1605.85	7141.80	1023355	4470080
Mean	42.67	4007.01	10620.15	2734844.50	6895923.00
Median	48.50	4370.02	10749.29	2980797.00	7030953.00
Maximum	49	6016.65	13967.35	4188140	9177210
$n = 200$					
Minimum	61	19423.57	75790.70	8102139	35472766
Mean	86.67	42275.21	112095.34	22585087.67	56542608.00
Median	98.00	44403.64	113046.62	24530982.50	57848377.00
Maximum	99	62789.41	143369.89	34915219	76304770

Acknowledgments

This research was partially supported by the Research Grants Council of Hong Kong, grant no. CUHK4214/01E.

References

1. A. Beck and M. Teboulle, "Global optimality conditions for quadratic optimization problems with binary constraints," SIAM Journal on Optimization, vol. 11, no. 1, pp. 179–188, 2000.
2. W. Conley, Computer Optimization Techniques, Petrocelli Books Inc.: New York, 1980.
3. M.W. Cooper, "A survey of methods for pure nonlinear programming," Management Science, vol. 27, no. 3, pp. 353–361, 1981.
4. M.W. Cooper, "Nonlinear integer programming for various forms of constraints," Naval Research Logistics Quarterly, vol. 29, no. 4, pp. 585–592, 1983.
5. L.C.W. Dixon, J. Gomulka, and S.E. Herson, "Reflection on global optimization problems," in Optimization in Action: Proceedings of the Conference on Optimization in Action held at the University of Bristol in January 1975, L.C.W. Dixon (Ed.), New York, 1976 pp. 398–435.
6. M.L. Fisher, "The Lagrangian relaxation method for solving integer programming problems," Management Science, vol. 27, no. 1, pp. 1–18, 1981.
7. R.-P. Ge, "A filled function method for finding a global minimizer of a function of several variables," Mathematical Programming, vol. 46, no. 2, pp. 191–204, 1990.
8. R.-P. Ge and C.-B. Huang, "A continuous approach to nonlinear integer programming," Applied Mathematics and Computation, vol. 34, no. 1, pp. 39–60, 1989.
9. R.-P. Ge and Y.-F. Qin, "A class of filled functions for finding global minimizers of a function of several variables," Journal of Optimization Theory and Applications, vol. 54, no. 2, pp. 241–252, 1987.
10. A.M. Geoffirion, "Lagrangian relaxation for integer programming," Mathematical Programming Study, vol. 2, pp. 82–114, 1974.
11. A.A. Goldstein and J.F. Price, "On descent from local minima," Mathematics of Computation, vol. 25, no. 115, pp. 569–574, 1971.
12. O.K. Gupta and A. Ravindran, "Branch and bound experiments in convex nonlinear integer programming," Management Science, vol. 31, no. 12, pp. 1533–1546, 1985.
13. Q.-M. Han and J.-Y. Han, "Revised filled function methods for global optimization," Applied Mathematics and Computation, vol. 119, nos. 2/3, pp. 217–228, 2001.
14. W. Hock and K. Schittkowski, Test Examples for Nonlinear Programming Codes, Springer-Verlag, New York, 1981.
15. F. Körner, "A new branching rule for the branch and bound algorithm for solving nonlinear integer programming problems," BIT, vol. 28, no. 3, pp. 701–708, 1988.
16. A.V. Levy and A. Montalvo, "The tunneling algorithm for the global minimization of functions," SIAM Journal on Scientific and Statistical Computing, vol. 6, no. 1, pp. 15–29, 1985.
17. D. Li and X.-L. Sun, "Success guarantee of dual search in integer programming: pth power Lagrangian method," Journal of Global Optimization, vol. 18, pp. 235–254, 2000.
18. D. Li and D.J. White, "pth power Lagrangian method for integer programming," Annals of Operations Research, vol. 98, pp. 151–170, 2000.
19. V.V. Litinetski and B.M. Abramzon, "MARS—A multi-start adaptive random search method for global constrained optimization in engineering applications," Engineering Optimization, vol. 30, pp. 125–154, 1998.
20. X. Liu, "Finding global minima with a computable filled function," Journal of Global Optimization, vol. 19, no. 2, pp. 151–161, 2001.
21. R. Luus and T.H.I. Jaakola, "Optimization by direct search and systematic reduction of the size of the search region," AIChE Journal, vol. 19, no. 4, pp. 760–765, 1973.
22. C. Mohan and 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, vol. 14, pp. 103–132, 1999.

23. J.J. Moré, B.S. Garbow, and K.E. Hillstrom, "Testing unconstrained optimization software," ACM Transactions on Mathematical Software, vol. 7, no. 1, pp. 17–41, 1981.
24. W.L. Price, "Global optimization by controlled random search," Journal of Optimization: Theory and Applications, vol. 40, pp. 333–348, 1983.
25. K. Schittkowski, More Test Examples for Nonlinear Programming Codes, Springer-Verlag, 1987.
26. X.-L. Sun and D. Li, "Asymptotic strong duality for bounded integer programming: A logarithmic-exponential dual formulation," Mathematics of Operations Research, vol. 25, pp. 625–644, 2000.
27. Z. Xu, H.-X. Huang, P.M. Pardalos, and C.-X. Xu, "Filled functions for unconstrained global optimization," Journal of Global Optimization, vol. 20, no. 1, pp. 49–65, 2001.
28. X.Q. Yang and C.J. Goh, "A nonlinear Lagrangian function for discrete optimization problems," in From Local to Global Optimization, A. Migdalas, P. M. Pardalos, and P. Värbrand (Eds.), Kluwer Academic Publishers, Dordrecht, 2001.
29. L.-S. Zhang, F. Gao, and W.-X. Zhu, "Nonlinear integer programming and global optimization," Journal of Computational Mathematics, vol. 17, no. 2, pp. 179–190, 1999.
30. L.-S. Zhang, C.-K. Ng, D. Li, and W.-W. Tian, "A new filled function method for global optimization," Journal of Global Optimization, vol. 28, no. 1, pp. 17–43, 2004.
31. Q. Zheng and D.-M. Zhuang, "'Integral global minimization: Algorithms, implementations and numerical tests," Journal of Global Optimization, vol. 7, no. 4, pp. 421–454, 1995.
32. W.-X. Zhu, "An approximate algorithm for nonlinear integer programming," Applied Mathematics and Computation, vol. 93, nos. 2/3, pp. 183–193, 1998.