PACKING EQUAL CIRCLES IN A DAMAGED SQUARE USING SIMULATED ANNEALING AND GREEDY VACANCY SEARCH

by

Xinyi Zhuang

B.Sc., Macau University of Science and Technology, 2011

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER SCIENCE

THE UNIVERSITY OF NORTHERN BRITISH COLUMBIA

Feb, 2015

Abstract

This thesis defines and investigates a generalized circle packing problem. called Packing Equal Circles into a Damaged Square (PECDS). We introduce a new heuristic algorithm that enhances and combines the Greedy Vacancy Search (GVS) and Simulated Annealing (SA), and demonstrate, through a series of experiments, its ability to find better solutions than either GVS or SA alone. The synergy between the enhanced GVS and SA, along with explicit convergence detection, makes the algorithm robust in escaping the points of local optimum.

Contents

List of Figures

List of Algorithms

List of Tables

Acknowledgement

Any Language cannot explain my infinite gratitude and appreciation to those who supported me with their love, believed my crazy and expensive idea of coming to Canada.

- My grandma, Xiuye Lin
- The woman who brought me to planet earth, Danrong You
- The wisest venture capitalist to any success in my current and future career, former member of Macau legislative council, Mr. Choikun Ung.
- The craziest and best buddy ever, Chao (Benjamin) Lin.
- The sweetest lady I had ever met, Chenwei Zhao.
- The most helpful senior in Prince George, Darlene Garand.
- The coolest buddy who installed four wheels under my feet, Guilin Xie.
- The kind and generous woman, Ms. Shuzhen Huang.
- The amazing buddies, Kin Leong, Yilong Yan.

Without any of these people, I cannot possibly sail through this journey and set foot on the wonderland that I had always dreamt about.

Also, huge thanks to both Dr. Desanka Polajnar and Dr. Jernej Polajnar who provided many intellectual insights, their suggestions on editing and articulation of the contents were amazing and contributed tremendously to the final version of this thesis.

Last but not least, thanks to Dr. Chen, who opened the door to Canada 10 minutes before my 23^{rd} birthday. The motivation of how one must always win in all ethical challenges, impervious to any temptation of political benefits, money, the convenience of easy reputation before pursuing any academic success is one of his great moral justices that influences me the most, he is truly a great mentor and a cool friend. I sincerely thank you all.

Chapter 1

Introduction

1.1 Overview

A packing problem is a problem of finding a minimum-size container of a specific shape that can hold a given number of identical objects of given shape and size without overlapping. For instance, one may seek the smallest square that contains a given number of identical circles of a given size; This problem is known as *Packing Equal Circles in a Squares (PECS)*[21]. Packing problems raises in many practical applications; e.g., PECS is associated with the packing tubes into rectangular crates, as well as with strategic placement of cellular towers. With the increasing number of objects, the number of valid packings increases exponentially and finding an optimal solution becomes computationally difficult. Many packing problems, including PECS are known to be NP-hard [48], which motivates research in heuristic algorithms.

The earliest reference to the problems of this class was recorded nearly 500 years ago, when the famous astronomer Johannes Kepler tackled the problem of finding the most efficient way to densely pack equal-sized spheres into a large crate without overlaps. He proposed a solution called the orange-pile arrangement [38], formally known as a face-centred cubic lattice. Later in the 19^{th} century, 350 years after the original announcement of the problem, German mathematician and physicist Carl Friedrich Gauss proved that Kepler's method was the most efficient solution for all "lattice packings". A lattice packing is one where the centres of the spheres are arranged in a regular 3-dimensional grid.

From the second half of the 20^{th} century, a number of Hungarian mathematicians have been trying to solve this problem by using computers. A website [44] created by Specht is dedicated to those pioneers. Since then, this field has been increasingly well studied by researchers. Many different heuristic algorithms have been proposed to find good solutions for the packing problems. These algorithms usually run in polynomial-time within their search space. Some of the popular models and strategies published in recent years are briefly explained in Chapter 2.

1.2 Contribution

In this thesis, I investigate heuristic algorithm for the equal circle packing problem where the square shape container may have a damaged interior. The damage is represented by a number of identical square-shape obstacles. I refer to this generalized version of PECS as *Packing Equal Circles in a Damaged Square (PECDS).* The heuristic algorithms that perform well on PECS are not necessary as effective on PECDS. In particular, our experiments demonstrate that Greedy Vacancy Search (GVS) algorithm [21] and the Simulated Annealing (SA) algorithm [24] exhibit significant inefficiencies when apply to some instances of PECDS.

The main contribution of this thesis is the introduction of a new heuristic algorithm for PECDS and an experimental demonstration of its ability to find better solutions than either GVS or SA. The new algorithm iterates an enhanced version of Greedy Vacancy Search algorithm followed by a modified Simulated Annealing algorithm, until the termination condition is met. The new algorithm is called *Enhanced Greedy Vacancy Search optimised by Simulated Annealing (eGVSXSA)*.

A key feature of eGVSXSA is that its termination is based on convergence detection (using fixed-size window of successive values of the objective function) rather than a fixed number of iteration. This principle applies to the inner loops within the GVSX and SA components as well as the main loop that iterates GVSX and SA. A consequence of this is that the running time of the algorithm varies, since in the absence of an imposed limit it keeps running as long as significant improvement are possible. The synergy of GVSX and SA, along with an explicit convergence detection, makes the algorithm robust in escaping the points of local optimum.

The remainder of this thesis is structured as follows: Chapter 2 briefly introduces three popular models for solving circle packing problems. Chapter 3 contains the problem statement and a list of definitions of terms frequently used in this thesis. Chapter 4 details the main algorithms eGVSXSA proposed in this thesis. Chapter 5 presents the results of the main experiments. The graphical representation of the experiment results is given in the appendix.

Chapter 2

Literature Review

In this chapter, I briefly review three popular models for solving circle packing problems published in the scientific literature. The review indicates in recent years, circle packing problems have been resolved by many different approaches [44]. These approaches are used to solve circles packing problems in various sizes of regular-shape container.

2.1 Linear Model for approximation

Galiev and Lisafina [13] proposed a linear model for packing equal circles in a domain G. The model was built upon a number of input arguments:

- A closed bounded domain G.
- Fixed circle radius r.

A finite set of points T is then constructed based on domain G. These points form a rectangular grid of size $\triangle$, $r = k \times \triangle, k \in \mathbb{Z}^+$, along both horizontal and vertical direction. The linearity of this model relies on the assumption that the center of the circles can only be placed on the grid.

The objective is to search the maximum number of non-overlapping circles placed in domain G and determine the center points of each circles. A feasible packing is evaluated by the weight levels c_i corresponding to individual circle position x_{ij} converted to z_i.

$$z_i = \begin{cases} 1 & x_{ij} \in T \\ 0 & otherwise \end{cases} \tag{2.1}$$

The measurement of a packing is denoted as N, given by the following equation.

$$N = \sum_{i=1}^{n} c_i \times z_i \tag{2.2}$$

The higher value of N is preferred and is achieved when more circles are on the grid ($z_i = 1$). Galiev and Lisafina [13] proposed a heuristic algorithm that approximates the maximum number of circles with the linear model and the performance of the algorithm depends on the selected $\triangle$. Their work demonstrated an instance of packing a number of circles with fixed radius into a rectangle in which two fixed circle areas can be considered as damages.

2.2 Quasi-Physical Method

In this model, each circle is considered as an elastic cylinder. The container edges are enforced by elastic springs. Overlaps among the circles and their container cause increment of the elastic energy. the accumulation of which indicates the overall energy. The size of the container is determined by the arrangement of the packing itself. The goal of the problem is to find a solution so that the objective function (or fitness function[23]) which depends on cumulative energy and size of container is minimized.

In 2011, Huang used the similar concepts to solve equal circle packing in a circular container and renewed over 40 best-known benchmarks (densely packed as well). The core algorithm for the global search is Basin-Hopping Search algorithm which simulates abrupt movements for the circles to escape from trapping in a local optimum. Although there are a number of scaling factors that they used with fixed values derived from computational experiments, the computational results proved the robustness of their proposed algorithm. In 2013, He et al. [16] further improved the concept of Quasi-Physical model to study Circle Packing Problem with Equilibrium Constraints (CPPEC) and generated 34 new CPPEC instances. Their methods used the overlapping depth represented by either elastic force or elastic energy as a measurement which is expected to be minimised. In Figure 2.1, the overlap is $AB = 0$ when $2 \times r - d_{ij} < 0$. A perfect global minimum when the energy represented by the overlaps and the size of the container are minimized.

A foundation method for Greedy Vacancy Search is BFGS (Broyden Fletcher Goldfarb Shanno algorithm for solving unconstrained optimization problem), also known as a Newton Downhill method which is used to search for the minimum of a function. I will not discuss the details of how BFGS [35, p. 194-201] and Limited-memory BFGS [35, p. 222-248], as it is not the purpose of this thesis. The substitution can be done if any good local optimization method such as Basin-Hopping Algorithm or improved Steepest Gradient Descent can produce better local optimum.

By using the unit test, BFGS(*fminunc*) implementation by MathWorks [31] compared to the two methods : BFGS implementation and Steepest Gradient Descent(SGD) developed by D.Kroon [12] gives better results as is shown in Figure 2.2. In this test, the same parameters are used in all methods. The unit test indicates that the BFGS by MathWorks [31] has the best overall performance. Despite the

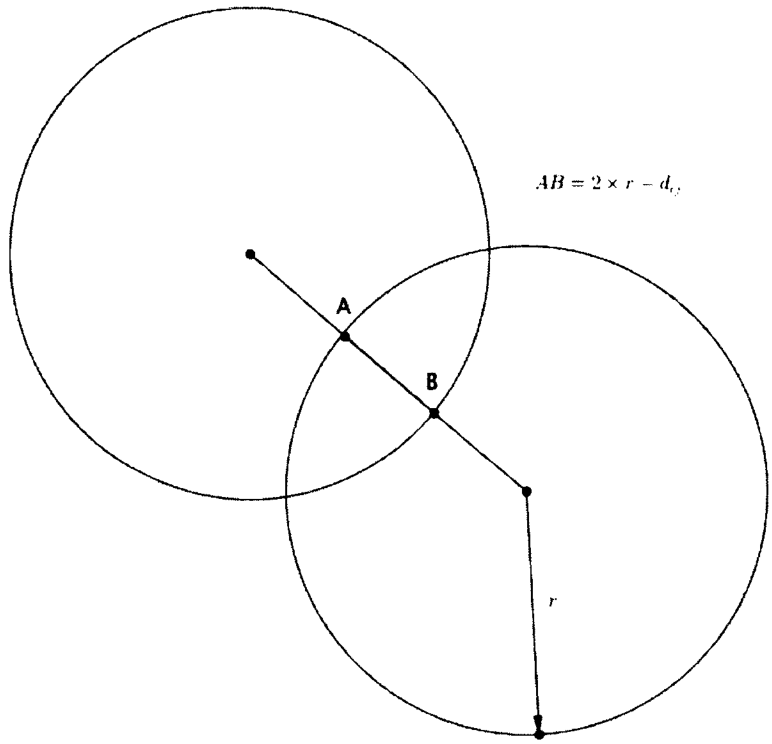


Figure 2.1: Overlapping Depth between two equal circles with radius r

fact that the BFGS by D.Kroon [12] found the minimum objective function value of all 100 iterations, the BFGS by MathWorks [31] has a more overall stability of performance. Therefore, I have adopted MathWorks [31]'s implementation.

2.3 Evolutionary Algorithms

One popular evolutionary algorithm is Genetic Algorithm, inspired by a biological evolution theory named *natural selection*. The term *natural selection* was first introduced by Charles Darwin in 1859 setting up one of the cornerstones of modern biology. Likewise, Genetic Algorithm inherits the main components of natural selection process such as inheritance, mutation, selection and crossover. The first generation offspring carries better information with respect to the fitness function. The mutation and crossover among the offspring require an explicit selection driven

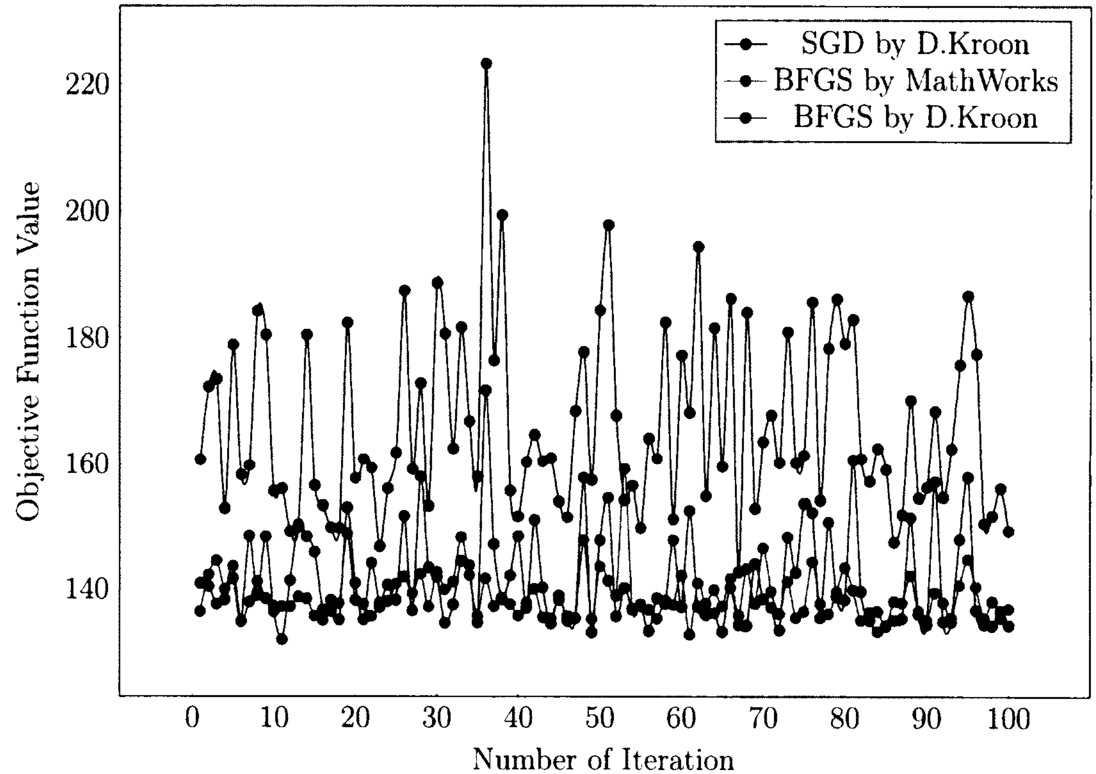


Figure 2.2: Performance statistics of 2 different BFGS implementations and SGD

by fitness function which evaluates the quality of evolution process for each new offspring. In order to use Genetic Algorithm to solve PECS problem, a linear representation of the solution is required. For this algorithm, each packing must be translated into an array of feasible types and structures that can evolve to a better packing. In 1996, Jakobs [23] manages to represent the packing pattern with a permutation in which the order of the position is generated by Bottom-Left strategy. This representation was effectively used to perform crossover and mutation process for unequal size rectangular packing. Their algorithm was then further extended to solve packing a number of unequal size of polygons in a rectangular container. This concept is further optimized by De-fu [11] in 2007 to solve strip rectangular packing problem.

Another popular algorithm is Simulated Annealing. The method simulates the annealing in metallurgy [25], a process of heating the material to its melting point, then slowly cooling down the temperature to reduce the defects, thus minimize the system's thermodynamic free energy and obtain a satisfying solution. In many circle packing problem with NP Complexity, this algorithm can often achieve good results [19, 46, 26, 51, 20]. The Simulated Annealing algorithm is based the neighbour searching mechanism, where the neighbours are generated based on the temperature and the current sate of solution constrained by an upper and lower bound. The temperature is represented by a parameter whose value decrease from 1 to 0.

The solution under higher temperature has higher probability to be accepted as the current solution in each iteration which often produces less optimized packing. In lower temperature, whether a solution is accepted or rejected depends on a probability and its objective function value.

Chapter 3

Problem Statement

In this chapter, I define the research problem addressed in this thesis. The inspiration for my research comes from studying the packing problem of equal circles in a regular shaped container, and the possibility to solve a more general version of the problem by introducing damaged square container.

3.1 Packing Equal Circles in a Damaged Square

To generalize the current problem of equal-radius circle packing in a regular shape container, I consider packing equal-radius circles in a damaged square container with the center of the container placed in the origin of the Cartesian Coordinate System. The damage square is a square whose interior contains randomly generated square obstacles. I define the damage areas by dividing the sides of the square container into n ($n \in \mathbb{Z}^+$) parts producing n^2 small squares. I then randomly select n' ($n' \in \mathbb{Z}^+$) squares to represent damaged areas. An example of randomly select 3 damaged regions out of 25 candidates is shown in Figure 3.1.

The problem can be defined in two ways depending whether we are looking at

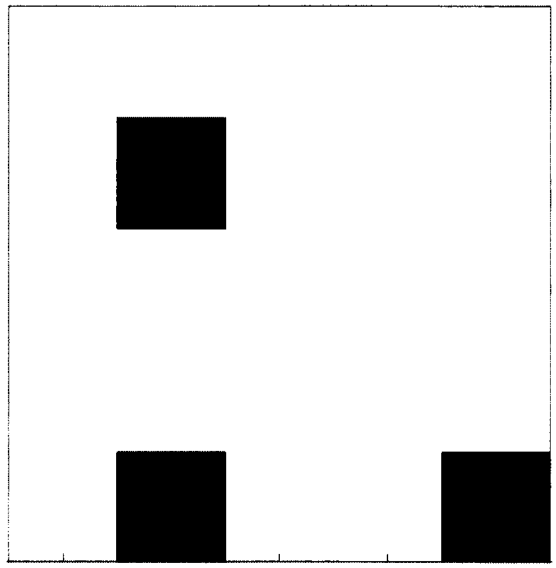


Figure 3.1: An example of damaged areas (distortions). $[3/5^2]$

packing unit circles into a minimum size square or packing a number of maximum size circles in a unit square. In both cases, the number of circles is given. In the following paragraphs, I formally define two versions of the problem: PECuS (Packing Equal Circles in a Unit damage Square) and PEuCS (Packing Equal unit Circles in a damaged Square).

- PECuS: Let the length of the damaged square container be $\mathbf{S} = 1$. The objective is to arrange $\mathfrak{N}$ equal circles with maximum radius r inside a damaged unit square without overlapping.

- PEuCS: Let the circle radius be $r = 1$. The objective is to arrange $\mathfrak{N}$ unit circles inside a minimum damaged square of size $\mathbf{S}$ without overlapping.

These two definitions describe the same problem of packing non-overlapping circles in a damaged square but with two different objectives. In PECuS we are optimizing the radius of circles while keeping unchanged container size and in PEuCS we are minimizing the size of the square container while keeping the radius of circles unchanged. However, it is possible to mutually convert these two solutions for the problem. This conversion can be done under the condition that the side-to-radius ratio in PECuS is equal to side-to-radius ratio in PEuCS.

Let us define the λ variable to be $\lambda = \frac{\mathbf{S}}{r}$. The definition of λ for PECuS with packing $\mathbf{P_{us}}$ is:

$$\lambda_{\mathbf{us}} = \frac{\mathbf{S_{us}}}{r_{\mathbf{us}}} = \frac{1}{r_{\mathbf{us}}} \tag{3.1}$$

For PEuCS with packing $\mathbf{P_{uc}}$, λ is:

$$\lambda_{\mathbf{uc}} = \frac{\mathbf{S_{uc}}}{r_{\mathbf{uc}}} = \frac{\mathbf{S_{uc}}}{1} \tag{3.2}$$

If $\lambda_{\mathbf{us}} = \lambda_{\mathbf{uc}}$ then the conversion can be calculated using the following equation:

$$\mathbf{P_{us}} = \frac{\mathbf{P_{uc}} - x_{bl} - \mathbf{S_{uc}}}{\mathbf{S_{uc}}} + 0.5 \tag{3.3}$$

where, x_{bl} is the horizontal coordinate of the bottom-left corner of the square container.

Figure 3.2 shows an example of converting a packing solution of 33 circles from solution space in PEuCS(left) to PECuS(right) using the above defined equations and conditions.

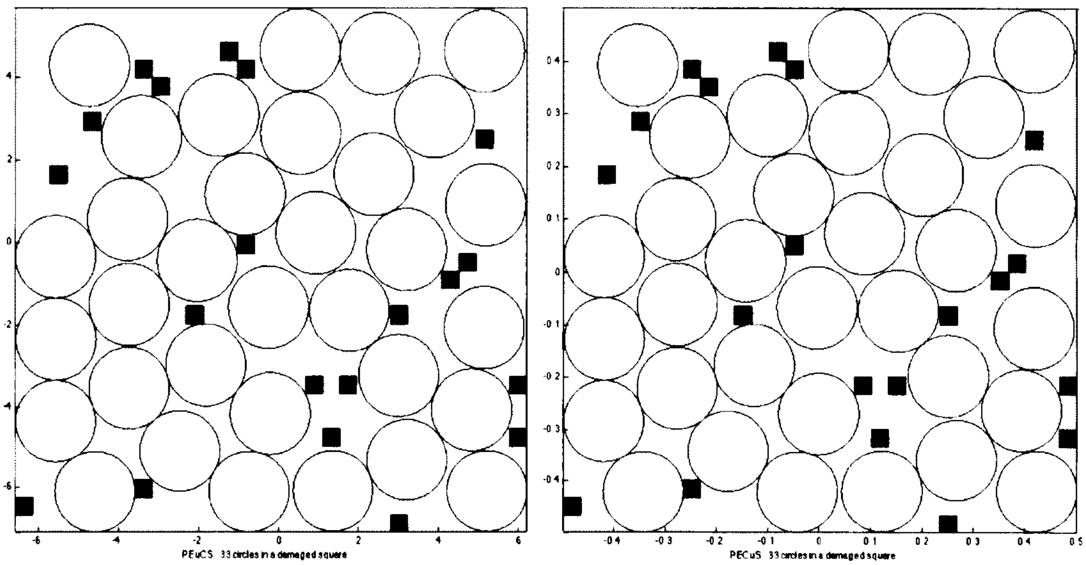


Figure 3.2: An example of converting optimal packing solution of 33 circles between solution space in PECuS and PEuCS.

3.2 Frequently used Terminology

To maintain the consistency of terms and symbols, let's define a number of frequently referred symbols and terms that are used throughout this thesis listed in Table 3.1, details regarding to what particular symbols stand for are explained under the description column.

Symbol	Description
$\mathbf{S}$	The side length of the square container.
r	The radius of a circle.
$\mathfrak{N}$	The number of circles, also known as the number of variables (in this thesis $\mathfrak{N} \geq 2$).
$\mathbf{P}'$	$\mathfrak{N} \times 2$ matrix of center coordinates; a feasible packing of non-overlapping circles in a square.
$\mathbf{P}$	$\mathfrak{N} \times 2$ matrix of center coordinates; a global optimal packing
H	Convergence value of the objective function.
PECuS	Packing equal circles in a damaged unit square.
PEuCS	Packing unit circles in a damaged square of size $\mathbf{S}$.
λ	Side-to-radius ratio which equals $\frac{\mathbf{S}}{r}$.
Θ	The upper bound of $\mathbf{S}$ for $\mathfrak{N}$ circles.
Ω	The lower bound of $\mathbf{S}$ for $\mathfrak{N}$ circles.
(x_i, y_i)	The center coordinates for circle i in Cartesian Coordinate System.
(x_{bl}, y_{bl})	The coordinates of the bottom left point of a square container in Cartesian Coordinate System.
(x'_{bl}, y'_{bl})	The coordinates of the bottom left point of a damage region inside the square container.
d_{ij}	The center distance between circle i and circle j.
f	The objective function.
$\mathbf{E}_\xi$	The cumulative overlapping depths between all the circles and the damaged regions.
$\mathbf{E}'$	Elastic energy, the cumulative overlapping depths between a new circle and an existing packing.
$\mathbf{E}$	Overall Energy, the cumulative overlapping depths of among all circles and the square container as well as the circles and damage regions
T	The temperature schedule for simulated annealing.
eps	Floating-point relative accuracy (3×10^{-12}) that specify the minimum overlapping tolerance.

Table 3.1: List of frequently used symbols in this thesis

Chapter 4

Proposed Solutions

In this chapter, I formulate the problem search space and propose the solutions for the problem of packing equal circles in a damaged square container.

4.1 Search Space Formulation

The search space of the packing problem of equal circles in a damaged square container consists of the center points of the given number of circles within the damaged square container of size $\mathbf{S}$ which is positioned in the center of the Cartesian Coordinate System. Next, I define a feasible packing $\mathbf{P}'$ of equal circles in a damaged container.

A feasible packing $\mathbf{P}'$ of $\mathfrak{N}$ ($\mathfrak{N} \in \mathbb{Z}^+$) non-overlapping unit circles in a damaged square container of size $\mathbf{S}$ is a packing with overall energy $\mathbf{E} = 0$. The damages are represented by a number of small square obstacles inside the container.

4.1.1 Feasible Packing

The feasible packing $\mathbf{P}'$ in the Cartesian Coordinate System is defined as:

$$\mathbf{P}' = [X, Y] = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ \dots & \\ x_{\mathfrak{N}} & y_{\mathfrak{N}} \end{bmatrix} \tag{4.1}$$

where $(x_i, y_i), i \in [1..\mathfrak{N}]$ is the center point of circle i.

Also, I introduce x_{bl} is the horizontal bottom left coordinate of the square container and y_{bl} is the vertical bottom left coordinate of the square container whose value are:

$$\begin{aligned} x_{bl} &= X_{min} - 1 \\ y_{bl} &= Y_{min} - 1 \end{aligned} \tag{4.2}$$

where, $X_{min} = min([x_1, x_2, ..., x_{\mathfrak{N}}]')$ and $Y_{min} = min([y_1, y_2, ..., y_{\mathfrak{N}}]')$.

Given a feasible packing solution $\mathbf{P}'$, the square size can be calculated by following equation:

$$\mathbf{S} = max(X_{max} - X_{min} + 2, Y_{max} - Y_{min} + 2) \tag{4.3}$$

In order for the circles to be inside the container, their center coordinates have to satisfy the follow conditions:

$$\begin{aligned} x_{bl} + 1 \leqslant x_i \leqslant x_{bl} + \mathbf{S} - 1 \\ x_{bl} + 1 \leqslant y_i \leqslant y_{bl} + \mathbf{S} - 1 \end{aligned} \tag{4.4}$$

The distance d_{ij} between circle i (x_i, y_i) and circle j (x_j, y_j) can be calculated as follow:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \geqslant d_{min} \tag{4.5}$$

where $i, j \in \mathbb{Z}^+$ and $i \neq j$, $d_{min} = 2 - eps$.

4.1.2 The Energy

The overall energy of a packing consists of all overlapping depths among the circles, circles and square container, and circles and damages.

The energy e_{ij} between circle i and circle j is defined as:

$$e_{ij} = \begin{cases} 2 - eps - d_{ij} & \text{if } 2 - eps - d_{ij} \geq 0 \\ 0 & \text{Otherwise} \end{cases} \tag{4.6}$$

where eps is the floating-point relative accuracy (3×10^{-12}) specifying the minimum overlapping tolerance and d_{ij} is the distance between circle i and circle j.

The energies e_{x_i} and e_{y_i} between circle i and the horizontal boundary, and circle i and vertical boundary are defined as:

$$\begin{aligned} e_{x_i} &= \begin{cases} x_{bl} + \frac{\mathbf{S}}{2} - |\frac{\mathbf{S}}{2} - 1| - x_i - eps & \text{if } x_{bl} + \frac{\mathbf{S}}{2} - |\frac{\mathbf{S}}{2} - 1| - x_i - eps \geq 0 \\ 0 & \text{Otherwise} \end{cases} \\ e_{y_i} &= \begin{cases} y_{bl} + \frac{\mathbf{S}}{2} - |\frac{\mathbf{S}}{2} - 1| - y_i - eps & \text{if } y_{bl} + \frac{\mathbf{S}}{2} - |\frac{\mathbf{S}}{2} - 1| - y_i - eps \geq 0 \\ 0 & \text{Otherwise} \end{cases} \end{aligned} \tag{4.7}$$

The third type of energy is between the circles and the damage region. The boundary of overlaps between a circle and a damaged region is shown in Figure 4.1. Each damage region is a square that can be denoted by the bottom-left coordinates (x'_{bl}, y'_{bl}) and its side length $s' = \frac{\mathbf{S}}{n}$. The boundary is a square shape with rounded corner with radius r. An overlap between a circle and the damage area occurs when the center coordinates of the circle are inside the boundary. An example of the boundary of the damaged region is shown in Figure 4.1.

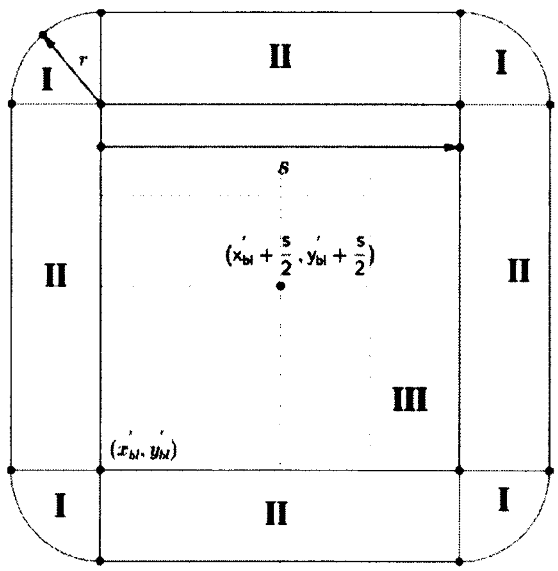


Figure 4.1: Boundary of overlaps between a circle and a damaged region

The overlap of a circle (x_i, y_i) and the damaged region (x'_{bl}, y'_{bl}, s') occurs when the center points satisfy the following equations:

$$
\begin{aligned}
d_x &= |x_i - (x'_{bl} + \frac{s'}{2})| < \frac{s'}{2} + 1 \\
d_y &= |y_i - (y'_{bl} + \frac{s'}{2})| < \frac{s'}{2} + 1
\end{aligned}
\tag{4.8}
$$

$$
d' = \sqrt{d_x^2 + d_y^2|} < \frac{s'}{\sqrt{2}} + 1 \tag{4.9}
$$

where d_x, d_y are horizontal and vertical distance between the center of circle i and the center point of a damage region respectively, d' is the center distance between circle i and the damage region.

The energy e_z between a circle and the four rounded corners of a damaged region and the energy e_z' between a circle and the horizontal/vertical boundary of a damaged region are defined as:

$$\begin{aligned} e_z &= \frac{s'}{\sqrt{2}} + 1 - d' \\ e_z' &= \frac{s'}{2} + 1 - max(d_x, d_y) \end{aligned} \tag{4.10}$$

where s' is the size of the small square.

The cumulative energy $\mathbf{E}_\xi$ of the overlapping depths between a circle and the damaged area is defined as:

$$\mathbf{E}_\xi = \begin{cases} e_z^2 & \text{if the circle is Area I (Fig 4.1)} \\ e_z'^2 & \text{if the circle is in Area II or III} \\ 0 & \text{else} \end{cases} \tag{4.11}$$

The energy $\mathbf{E}$ of all overlaps is defined as:

$$\mathbf{E} = \sum_{i=1}^{\mathfrak{N}-1} \sum_{j=i+1}^{\mathfrak{N}} e_{ij}^2 + \sum_{i=1}^{\mathfrak{N}} (e_{x_i}^2 + e_{y_i}^2) + \sum_{i=1}^{\mathfrak{N}} \sum_{j=1}^{n'} \mathbf{E}_{\xi_{ij}} \tag{4.12}$$

where, n' is the number of damage regions.

In next section, I define the objective function for the problem.

4.1.3 Objective Function

To evaluate the feasible packing $\mathbf{P}'$, I define the objective function as:

$$f(\mathbf{P}') = \mathbf{S}^2 + \varphi \times \mathbf{E} \tag{4.13}$$

where $\mathbf{S}$ is the size of the damaged square container, φ is the penalty parameter, and $\mathbf{E}$ is the energy.

4.2 The search methods

In this section, I introduce three search algorithms, Local Search, Enhanced Greedy Vacancy Search, Simulated Annealing, that are used in solving the stated problem (Chapter 3)

4.2.1 Local Search and Convergence Detection

The local search algorithm is based on the quasi-downhill method such as BFGS for finding local optimum, in order to produce the initial feasible packing $\mathbf{P}'$. This algorithm first uses BFGS to find a packing that minimize the objective function $f(\mathbf{P}')$ (Eq (4.13)), then minimize the energy function $\mathbf{E}$ (Eq (4.12)) to find a feasible packing $\mathbf{P}'$.

There are many ways to detect the convergence of the objective function. I propose a runtime monitoring mechanism to determine the termination condition for the search method, namely *convergence detection*. This mechanism keeps a historical record of the current best objective function values found by any heuristic search algorithm (i.e. GVS or SA). The termination condition is determined by the difference between the first and last obtained values record, i.e stop when this

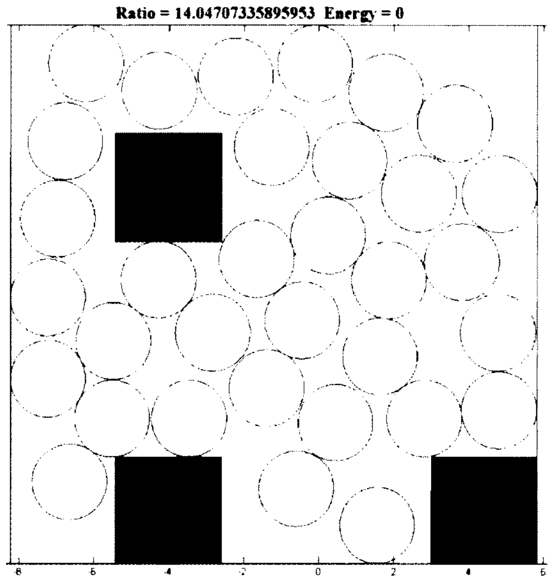


Figure 4.2: An example of finding a feasible packing with local search algorithm

difference is lower than eps. An example of feasible packing 33 unit disks is shown in Figure 4.2.

To detect the convergence: $h \nrightarrow H$, I use a queue Q of size m to record the objective function value of the currently found best packing $\mathbf{P}'$. $Q(i) = f(\mathbf{P}')$.

$$H = Q(1),\ if\ |Q(1) - Q(m)| \leq eps \tag{4.14}$$

The Local Search algorithm is given in Algorithm 4.1.

Algorithm 4.1 Local Search Algorithm

Input: $\mathfrak{N}$ circles

Output: A local optima $\mathbf{P}'$.

function LOCALSEARCH($\mathfrak{N}$)
 $s \leftarrow \mathfrak{N} * 2$
 $\varepsilon \leftarrow \infty$
 while $h \nrightarrow H$ **do**
  $p \leftarrow$ random center points between $-\frac{s}{2}$ to $\frac{s}{2}$
  $(p, s) \leftarrow \mathrm{BFGS}(f, p)$
  if $\mathbf{E}(p) > eps$ **then**
   $(p, s) \leftarrow \mathrm{BFGS}(E, p)$
  end if
  $h \leftarrow f(p)$
  if $h \leq \varepsilon$ **then**
   $\mathbf{P}' \leftarrow p$
   $\varepsilon \leftarrow h$
  end if
 end while
 Return $\mathbf{P}'$
end function

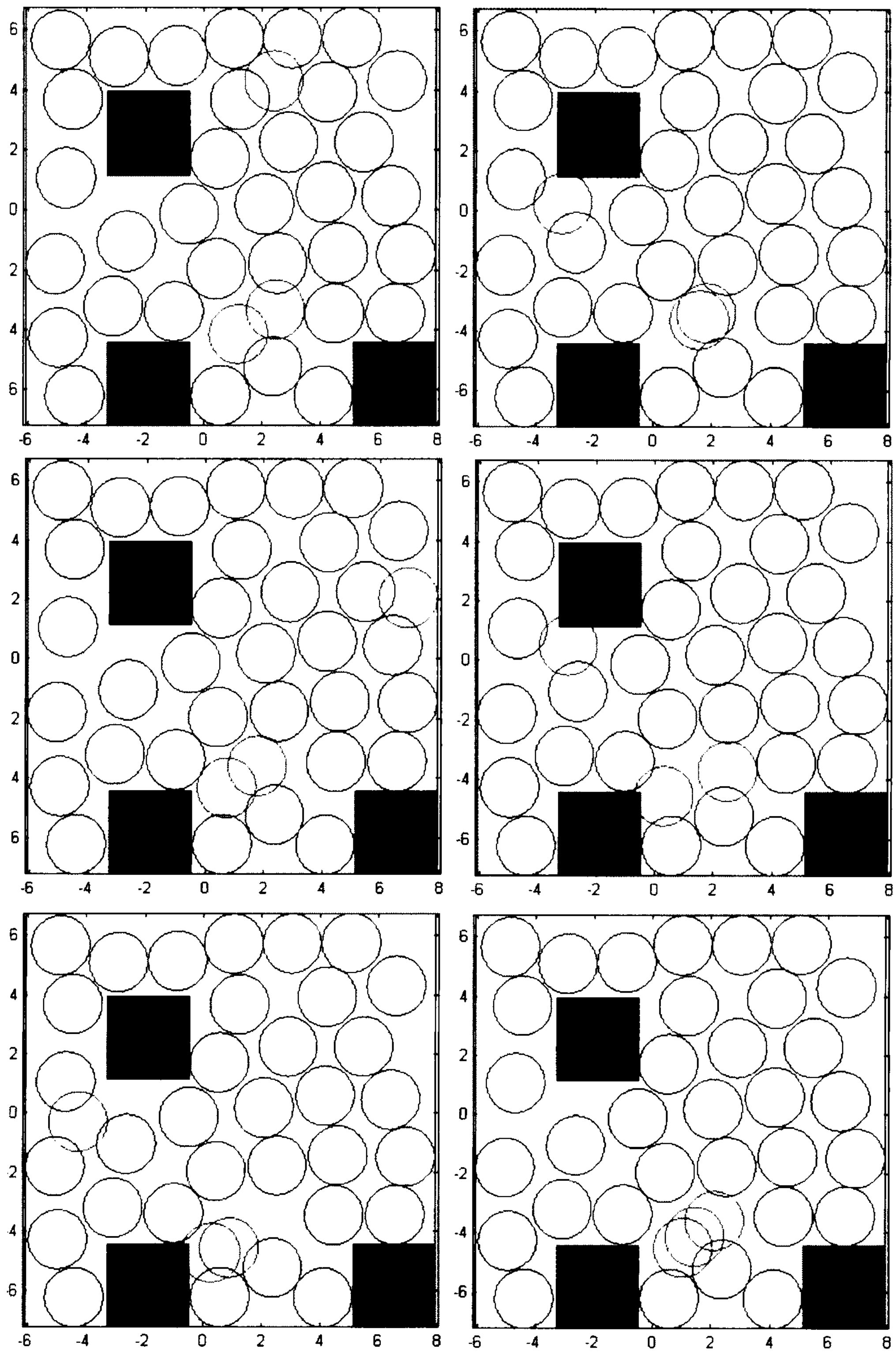
4.2.2 Enhanced Greedy Vacancy Search (GVSX)

The Greedy Vacancy Search (GVS) algorithm was first introduced by Huang and Ye [21]. The ways GVS works, is by fixing the size of the square container at a relatively large initial value, rearranging the current packing to the most vacant area at each iteration and then minimizing the energy of the packing inside the fixed square.

A candidate packing with lowest energy for the current container size is then chosen and passed to their local search procedure. The calculation of the container size of the packing with minimum energy is done in the local search. If the calculated size of the container is less than the current container's size, then the current container size is updated. The algorithm stops when the specified running time for the algorithm is reached. The run time depends on the given number of circles.

The original GVS has been proven to generate good results due to the physical model of the packing problem. The model assumes that the surfaces of the circles and the damaged container are perfectly smooth (coefficient of friction = 0). According to the *First Law of Friction*, the friction between any two surfaces is strictly proportional to the pressure between them. In other words, the friction is only caused by the spatial movement among the circles and between the circles and the damaged container. Using this model, the Vacancy Search (Algorithm 4.2) finds the biggest vacant area and relocates one of the circles to that area. The initial size of the relocated circle is set to be infinitely small and then increases the radius to the limit of 1. This will cause spatial movements of all the circles in the container. If this circle successfully reaches to its radius limit, a new local minimum is created. Otherwise, the container has to be enlarged as the radius of that circle increases and the overall movement all the circles. If this process is repeated enough, the better packing can be found. This turns out, as Huang and Ye [21] explains, to be the a series of deterministic mutation operations. An example of finding the vacancy

Figure 4.3: Example of apply vacancy search in a packing. (covered by red circles)



areas in a local optimum is shown in Figure 4.3.

The algorithm of finding the biggest vacant area can be seen in Algorithm 4.2. The approximate size of any vacant area is inversely proportional to the size of its elastic energy $\mathbf{E}'$. The elastic energy is higher when the circle is placed in a smaller vacant area that causes more overlaps. This is used to find the biggest vacant area. The elastic energy is an energy generated by relocation of a circle to the vacant area. This can be calculated in the follow way:

$$\mathbf{E}' = \sum_{i=1}^{\mathfrak{N}-1}(e_c^2 + e_x^2 + e_y^2) \tag{4.15}$$

where, e_c are the overlapping depths between the modified circle and other circles, e_x an e_y are overlapped to the vertical and horizontal boundary of the container respectively.

Algorithm 4.2 Most Vacant Area Search Algorithm

Input: A (preferably feasible) packing p of $\mathfrak{N}$ circles.
Output: The center coordinates of the most vacant area C.

function FINDVACANCY(p)
  $c \leftarrow$ Randomly scatter $3 \times \mathfrak{N}$ circles inside container of p
  $e_{min} \leftarrow inf$
  for $i \leftarrow 1$ *to* $3 \times \mathfrak{N}$ **do**
    $e \leftarrow \mathbf{E}'(c(i,:), p)$ ▷ Evaluate elastic energy of $c(i,:)$ over p
    if $e < e_{min}$ **then**
      $e_{min} \leftarrow e$
      $C \leftarrow c(i,:)$ ▷ Assign the i^{th} circle to C
    end if
  end for
  Return C
end function

In order for the original Vacancy Search to work for our problem, the elastic energy has to include the energy between the relocated circle and the damaged areas. The elastic energy between the relocated circle and one damage region is

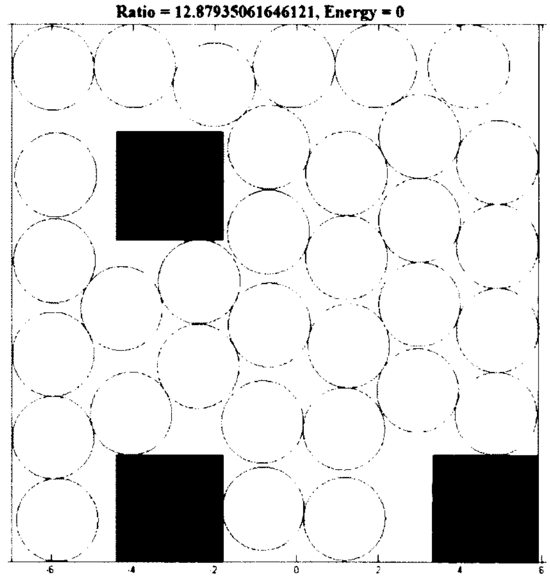


Figure 4.4: Optimising 33 dense packing circles with GVSX

denoted by $\mathbf{E}_\xi$. Thus the elastic for our problem is defined as followed:

$$\mathbf{E}' = \sum_{i=1}^{\mathfrak{N}-1}(e_c^2 + e_x^2 + e_y^2) + \sum_{j=1}^{n'} \mathbf{E}_\xi \tag{4.16}$$

where n' is the number of damaged regions.

When considering the original GVS method for solving the problem of packing equal circle in a damaged container, I noticed the possible issues related to the termination condition of the algorithm and their search criteria for finding the optimum solution. The termination criteria for this algorithm is a predefined runtime which depends on the number of packing circles. As for the issue for search criteria, by fixing the size of the container to search for a feasible packing does not guarantee the right solution due to the termination criteria. The algorithm may not converge during the given runtime. To avoid these two issues, an adaptive vacancy search algorithm named Enhanced Greedy Vacancy Search (GVSX) is proposed and shown in Algorithm 4.3.

In GVSX, the convergence detection proposed in Section 4.2.1 is adopted to solve the convergence problem. The search criteria problem is resolved by using a multi-objective search which iteratively minimises the objective function and if necessary the energy to find a local minimum. By using this criteria, the algorithm explore more possible local optimum than the original implementation and always terminates.

The implementation of GVSX adopts the Local Search (Algorithm 4.1) and Vacancy Search (Algorithm 4.2). It is an iterative process that takes an initial packing, relocate one of the circles to the biggest vacant area founded by Vacancy Search and then use Local Search to perform "downhill climbing". At each iteration, the current best packing is updated if a better local minimum (evaluated using our objective function, Eq (4.13)) is found and passed to the upcoming iteration until the objective function converges. An example of optimizing 33 dense packing circles by GVSX is shown in Figure 4.4.

Algorithm 4.3 Enhanced Greedy Vacancy Search Algorithm (GVSX)

Input: A feasible packing of $\mathfrak{N}$ circles x_0.
Output: An optimal packing $\mathbf{P}'$.

```
function GVS(x_0)
    s ← N * 2                          ▷ Initial side of square
    p ← x_0                            ▷ p is the current testing solution
    ε ← f(p)                           ▷ ε is the current best objective function value
    h ← inf                            ▷ h is the objective function value of the current testing solution
    i ← 1
    while h ↛ H do                     ▷ While h does not converge
        p(i) ← FindVacancy(p)          ▷ Find and relocate circle i to the most vacant area
        p ← BFGS(f, p)                 ▷ Minimize objective function
        if E(p) > eps then
            p ← BFGS(E, p)             ▷ Minimize energy function
        end if
        h ← f(p)
        if h ≤ ε then
            P' ← p                     ▷ Update the current best solution
            ε ← h
        end if
        i ← i + 1
        if i > 𝔑 then
            i ← 1                      ▷ Reset current index
        end if
    end while
    Return P'
end function
```

4.2.3 Simulated Annealing

The idea of Simulated Annealing (SA) search method comes from the simulation of the annealing process of iterative heating and cooling solids. The materials are heated up by increasing the temperature to a very high value, followed by a slow cooling process to lower the temperature such that the molecules of the annealing material are able to better arrange themselves in a low energy state. The standard SA algorithm has a temperature variable to simulate the heating process. This variable has a high initial value and then slowly decreases as the algorithm iterates. In each iteration, an equal number of the solution points are randomly generated constrained by the given upper and lower bounds as well as the current temperature. These points are evaluated in comparison with the current solution by the objective function (less is better). There are two different conditions to choose the current best solution. The first is to evaluate the objective function of the current test solution and choose the one with smaller value of the objective function than the current best solution. The second condition is taking a probability of accepting the current test solution regardless whether it is better than the current best solution. The second condition represents the re-heating process of annealing.

The most important contribution that Simulated Annealing(SA) provides to the solution of the stated problem is its nature of searching for as many local minimums as possible with sufficient amount of different initial guesses. It selects the best local minima as the global optimal. SA is very capable of finding a good solution for bound-constrained global optimization problem although it doesn't guarantee to yield a proven global optimum, it often finds satisfying solutions.

To demonstrate how original SA works, I use SA to optimize 33 circles packing in a damaged square container(Figure 3.1). As the temperature is scheduled from maximum 1.0 to minimum 0.0, 6 internal results are extracted in descending temperature, these results demonstrate the gradual movement of the current best packings, which reflects how SA successively obtains a better packing. As shown in Figure 4.5

SA starts with an initial guess array of points p_0, the same dimension array lower bound Ω and upper bound Θ [5], maximum iteration ℓ and function tolerance $\hbar$ (default value is 10^{-4}). At each iteration, a number of new testing points are randomly generated (denoted p_1) using uniform random vector transformed by the inverse μ-law [50, p.334–337]. These points must be constrained by the upper bound(Θ) and lower bound(Ω) in order to become an eligible guess. Essentially what each iteration does, it shifts the current points p within the bounds by $\triangle p$ generating $p_(1)$ as the new guessing points. These points ($p_1 = p + \triangle p$) are then taken as the current points (better solution) if they result in negative arousal ($\triangle h < 0$) to the objective function.

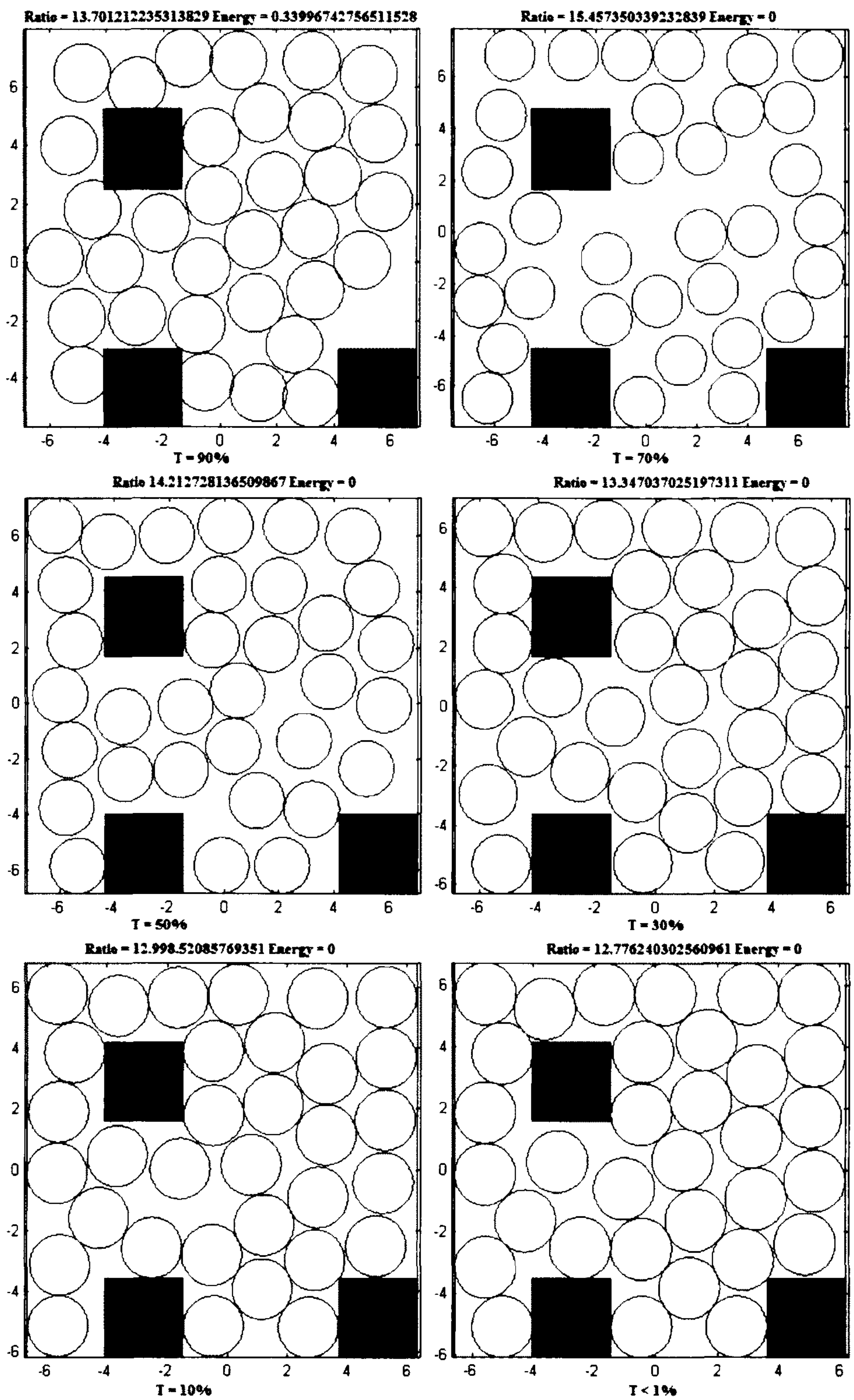
Algorithm 4.4 describes the implementation of the SA where array p represents the packing solutions. The SA method is modified based on the original implementation developed by Corte [8]. The modifications consists of applying my proposed convergence detection to stop the algorithm, dynamically updating the upper and lower bounds, and fixing the temperature to be .0 (0%).

The important aspect of this algorithm is fixing the temperature to .0 which causes the acceptance probability to .30. This probability can be determined by the following equation:

$$\rho(\triangle p) = e^{-T \times \dfrac{\triangle h}{|f(p) \times \hbar|}}, \quad for \triangle h > 0 \tag{4.17}$$

where $T = \dfrac{m}{\ell} = \dfrac{\ell}{\ell} = 1$, $f(p)$ is the objective function value of p, and $\hbar$ is the function tolerance $\hbar$ (default value is 10^{-4}).

Figure 4.5: Optimising 33 dense packing circles with Simulated Annealing



$\rho(\triangle p)$ remains closely to e^{-1} for $|\frac{\triangle h}{f(p)}| = \hbar$, which means that when the temperature cools down to 0, the probability of escaping the local minima by increasing the objective function with the value $\triangle h = |f(p)| \times \hbar$ remains 30%. This explains why the inverse temperature is used in the implementation of SA.

Algorithm 4.4 Modified Simulated Annealing with Convergence Detection

Input: Any packing p_0 of $\mathfrak{N}$ circles
Output: A feasible packing $\mathbf{P}'$.

function SA(p_0)
$p \leftarrow p_0$ ▷ p is current point, p_0 is current solution
$h_p \leftarrow f(p)$
$h_0 \leftarrow h_p$
$m \leftarrow 1$
while $h_0 \not\twoheadrightarrow H$ **do**
if $\mathbf{E}(\mathbf{P}') \leq eps$ **then**
$(\Omega, \Theta) \leftarrow$ half the size of the current container of $\mathbf{P}'$
end if
$T \leftarrow m/\ell$ ▷ T is calculated as inverse of temperature, from 0 to 1
if $T > 1$ **then**
$T \leftarrow 1$
end if
$m_u \leftarrow 10^{T*100}$
for $k \leftarrow 0$ *to* κ **do** ▷ κ is the max number of guess points,default 1000
$y \leftarrow$ random center points of $\mathfrak{N}$ circles
$\triangle p \leftarrow ((((1 + m_u).^{|y|} - 1)/m_u). * sign(y)). * (\Theta - \Omega)$
▷ .* is dot product in matrix notation
$p_1 \leftarrow p + \triangle p$ ▷ p_1 is current test point
$p_1 \leftarrow (p_1 < \Omega). * \Omega + (\Omega \leq p_1). * (p_1 \leq \Theta). * p1 + (\Theta < p1). * \Theta$
▷ Keep solution within bounds
$h_1 \leftarrow f(p_1)$
$\triangle h \leftarrow h_1 - h_p$
if $\triangle h < 0$ **or** $rand < e^{-T \times \frac{\triangle h}{(|h_p| + eps) \times \hbar}}$ **then**
$p \leftarrow p_1$
$h_p \leftarrow h_1$
end if
if $h_1 < h_0$ **then**
$\mathbf{P}' \leftarrow p_1$
$h_0 \leftarrow h_1$
end if ▷ $\hbar$ is function tolerance
end for
$m \leftarrow m + 1$
end while
Return $\mathbf{P}'$
end function

4.3 Main Algorithm : eGVSXSA

In this section, I define the main algorithm *Enhanced Greedy Vacancy Search optimised by Simulated Annealing (eGVSXSA)*(Algorithm 4.5), for solving the stated problem. The eGVSXSA utilizes the unique capability of Simulated Annealing in a lower energy state to enhanced GVSX in solving circle packing in various damaged containers. The results are encouraging and robust. In the previous chapter, we have introduced 3 methods: Local Search, GVSX and SA. Each of these three algorithms has its unique way of minimising the objective function.

Algorithm 4.5 Main Algorithm : eGVSXSA

Input: $\mathfrak{N}$
Output: A global optimal **P**.
function eGVSXSA($\mathfrak{N}$)
    $\varepsilon \leftarrow inf$ ▷ Initial value of current best objective function value
    $p_{ls} \leftarrow$ LocalSearch($\mathfrak{N}$) ▷ Get the initial local optimum
    while $h \nrightarrow H$ **do** ▷ while h does not converge
        $p_{gvs} \leftarrow \text{GVSX}(p_{ls})$ ▷ Optimize local optimum using GVSX
        $p_{sa} \leftarrow \text{SA}(p_{gvs})$ ▷ From lower state (0%) of temperature.
        $h \leftarrow f(p_{sa})$
        if $h \le \varepsilon$ **then**
            $\mathbf{P} \leftarrow p_{sa}$
            $\varepsilon \leftarrow h$
        end if
    end while
    Return P
end function

The representation of the damages in the container is denoted by $[n'/n^2]$, where n^2 is the number of equal squares from which n' squares are randomly selected as damages. For example, $[20/30^2]$ means that the side of the container is divided by 30 generating 900 equal squares from which 20 are randomly selected as damages.

The eGVSXSA first uses Local Search to obtain a initial local optimum, then applies GVSX to escape the newly generated local optimums. Finally, it uses SA

Table 4.1: 33 circles with $[3/5^2]$ by Local Search, GVS, and SA

	eGVSXSA	Local Search	GVS	Simulated Annealing
Ratio($\lambda = \frac{\mathbf{S}}{r} = \mathbf{S}$)	12.69633	14.04707	12.879351	12.77624
Radius($\frac{1}{\lambda}$)	0.078763	0.071189	0.077644	0.078270
Energy($\mathbf{E}$)	0.000000	0.000000	0.000000	0.000000

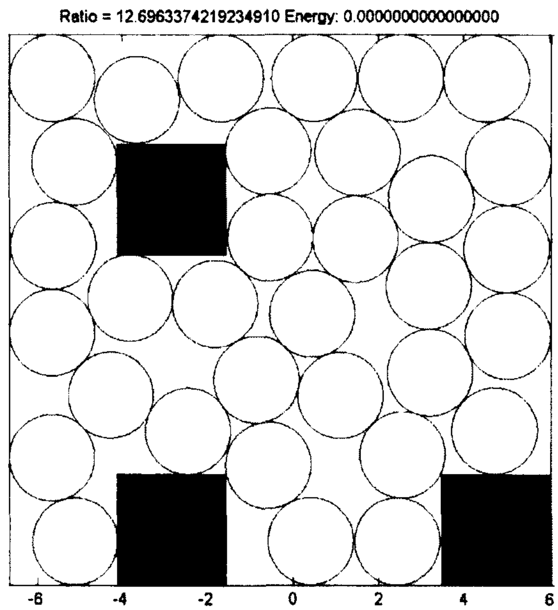


Figure 4.6: Better 33 circle packing in a damaged square $[3/5^2]$ found by eGVSXSA.

to optimize the solution from GVSX. This process is iterated until the convergence condition is reached. An example of using eGVSXSA is shown in Figure 4.6. Table 4.1 shows the comparison of the eGVSXSA with Local Search as well as the original GVS and SA.

Chapter 5

Experimental Results

In order to test the performance of GVSX and eGVSXSA in comparison with the original GVS and SA, we have conducted an experiment with two packings, 69 and 70 equal circles in a damaged square. The damages are randomly generated. The number of divided regions for selecting the damages is 900, and the number of selected damages is set to be 20, 30 and 40 respectively.

The tests are conducted in MATLAB on Windows 7 64-bit Operation System. The Process specification is 2 CPUs of *Intel(R) Xeon(R), CPU E5-2603 0 @1.80GHz*. The runtime of the tests are shown in Table 5.3.

I conducted two types of experiments. In the first experiment, the number of circles to be packed is 69 and 70 while the number of damages ranges from 20 to 40 (increment of 10). In the second experiment, the number of damages is 20 while the number of circles ranges from 30 to 68. The results demonstrate that as the amount of damages increases, eGVSXSA suffers the least impact and is able to search much smaller ratio while keeping the energy at zero. The graphical results are shown from Figure 5.2 to Figure 5.6. The numerical results of the second type of experiments are shown in Table 5.2 and the graphical results are shown in Figure 5.7.

Pairs	P-value
SA vs eGVXSA	3.62593×10^{-14}
SA vs GVSX	0.243103314
GVS vs SA	0.006631256
GVS vs eGVXSA	2.084685×10^{-14}
GVS vs GVSX	0.008093876
GVSX vs eGVXSA	1.32381×10^{-11}

Table 5.1: Significant test of objective function value (One tailed distribution, two-sample unequal variance, significance level : 5%)

The results also indicate that the original Simulated Annealing has better performance than the original GVS. This experiment indicates that GVSX and Simulated Annealing have overall better performance than the original GVS under the same amount of damages, while eGVSXSA has the best performance of all. This statement is confirmed by the significant test (shown in Table 5.1), where P-value is a function of the observed sample data set used for testing null hypothesis.

Figure 5.1: Experimental Result: 69 circle packing in a damaged square, $[20/30^2]$.

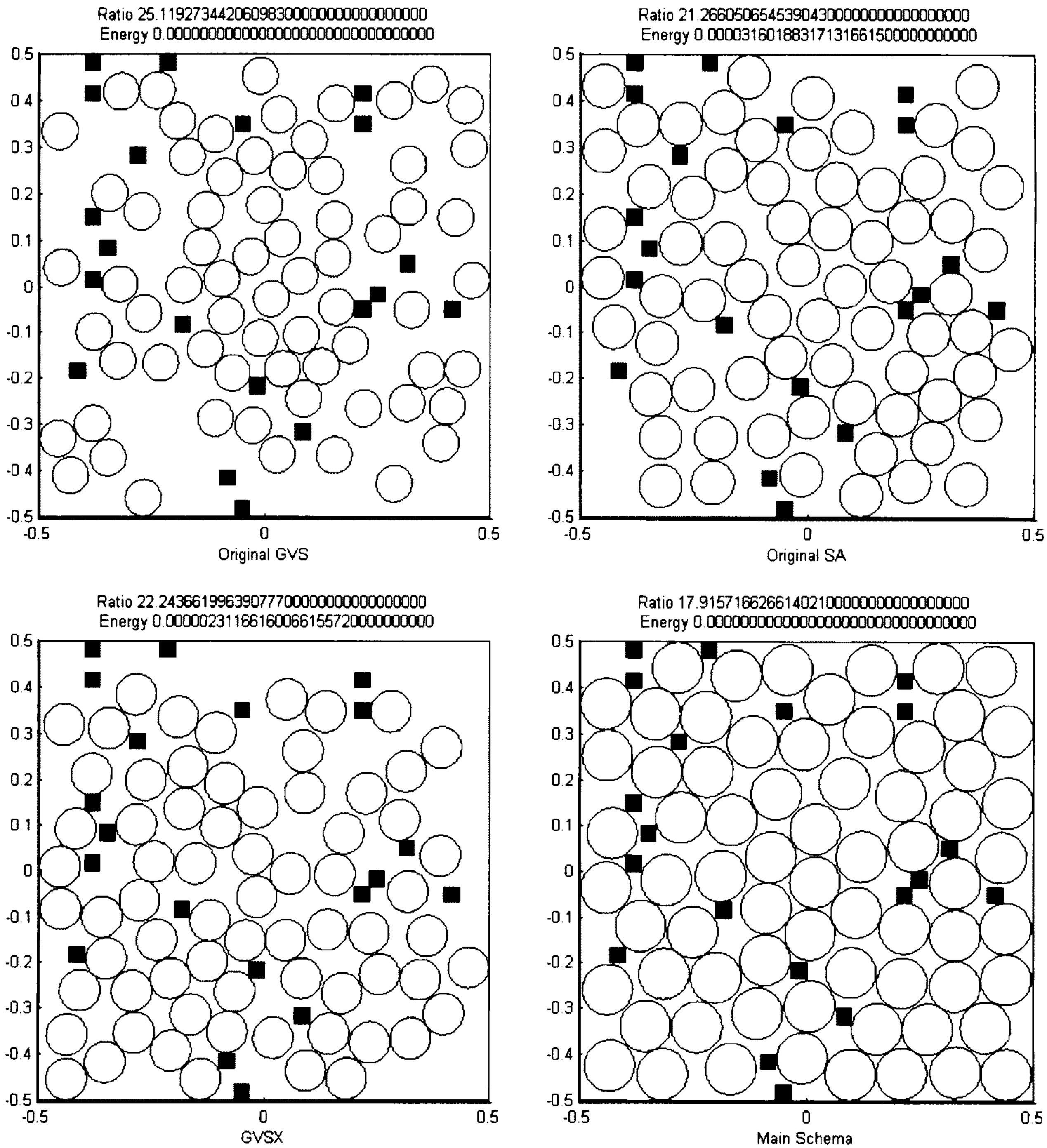


Figure 5.2: Experimental Result: 70 circle packing in a damaged square, $[20/30^2]$.

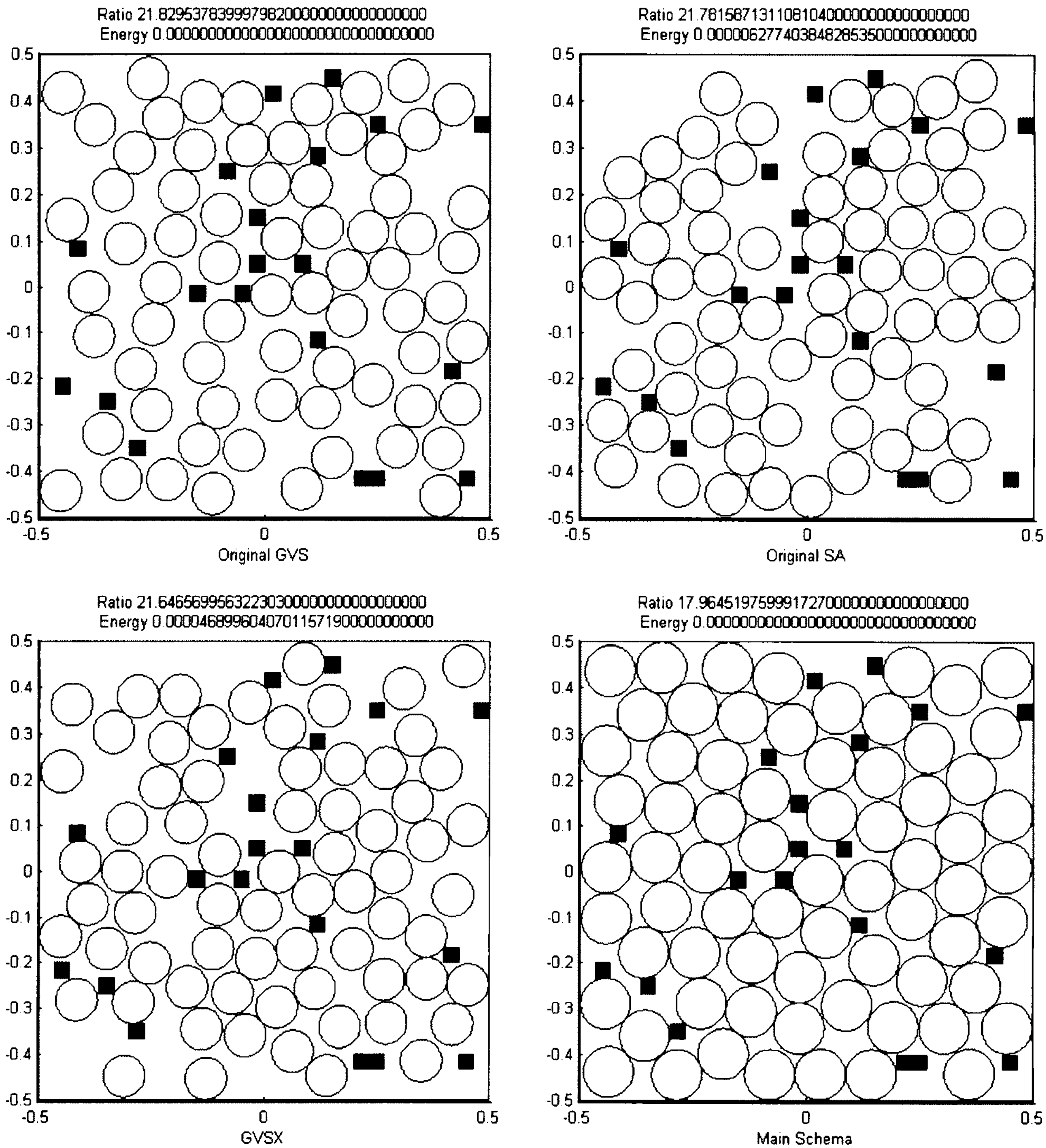


Figure 5.3: Experimental Result: 69 circle packing in a damaged square, $[30/30^2]$.

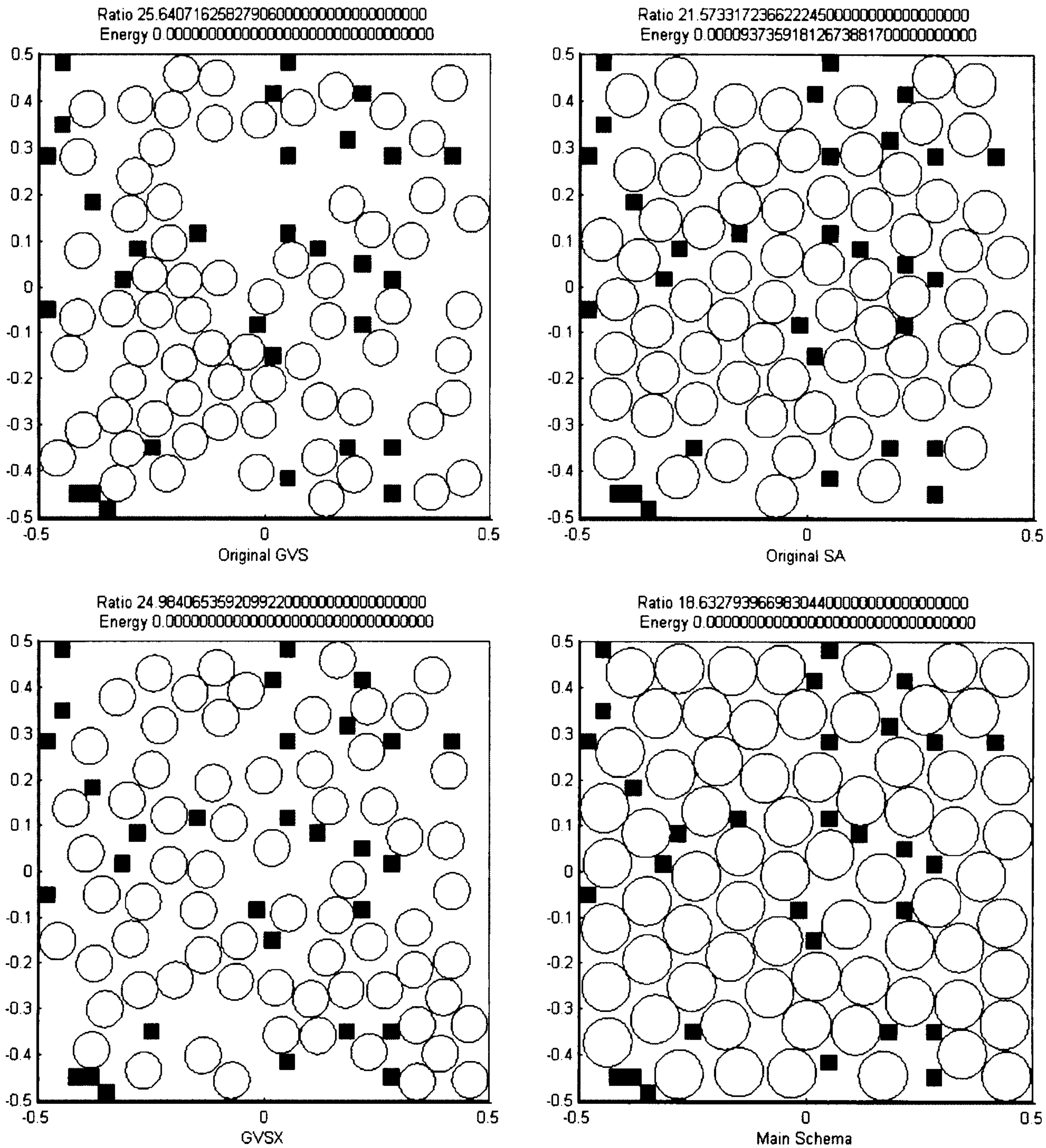


Figure 5.4: Experimental Result: 70 circle packing in a damaged square, $[30/30^2]$.

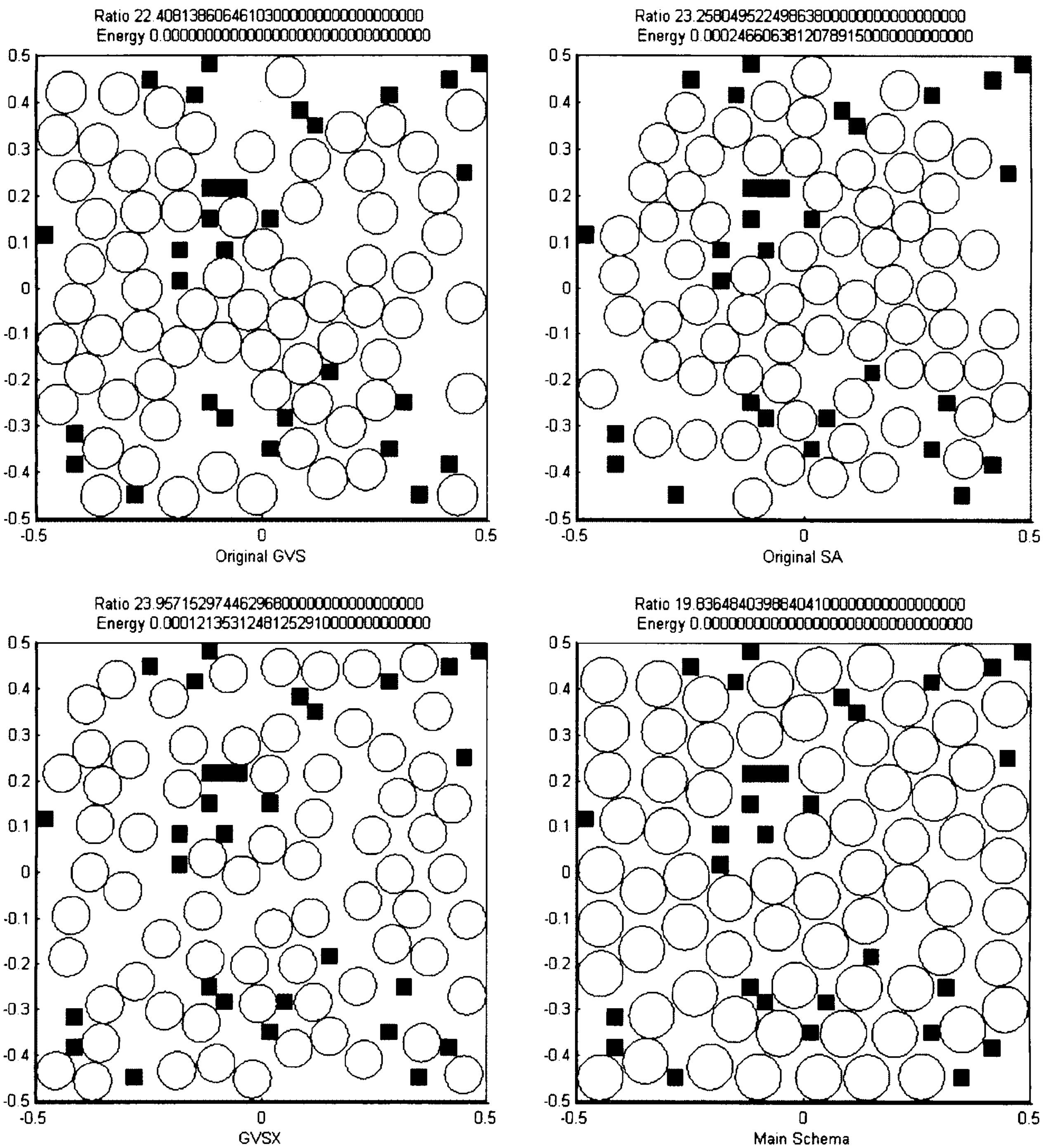


Figure 5.5: Experimental Result: 69 circle packing in a damaged square, $[40/30^2]$.

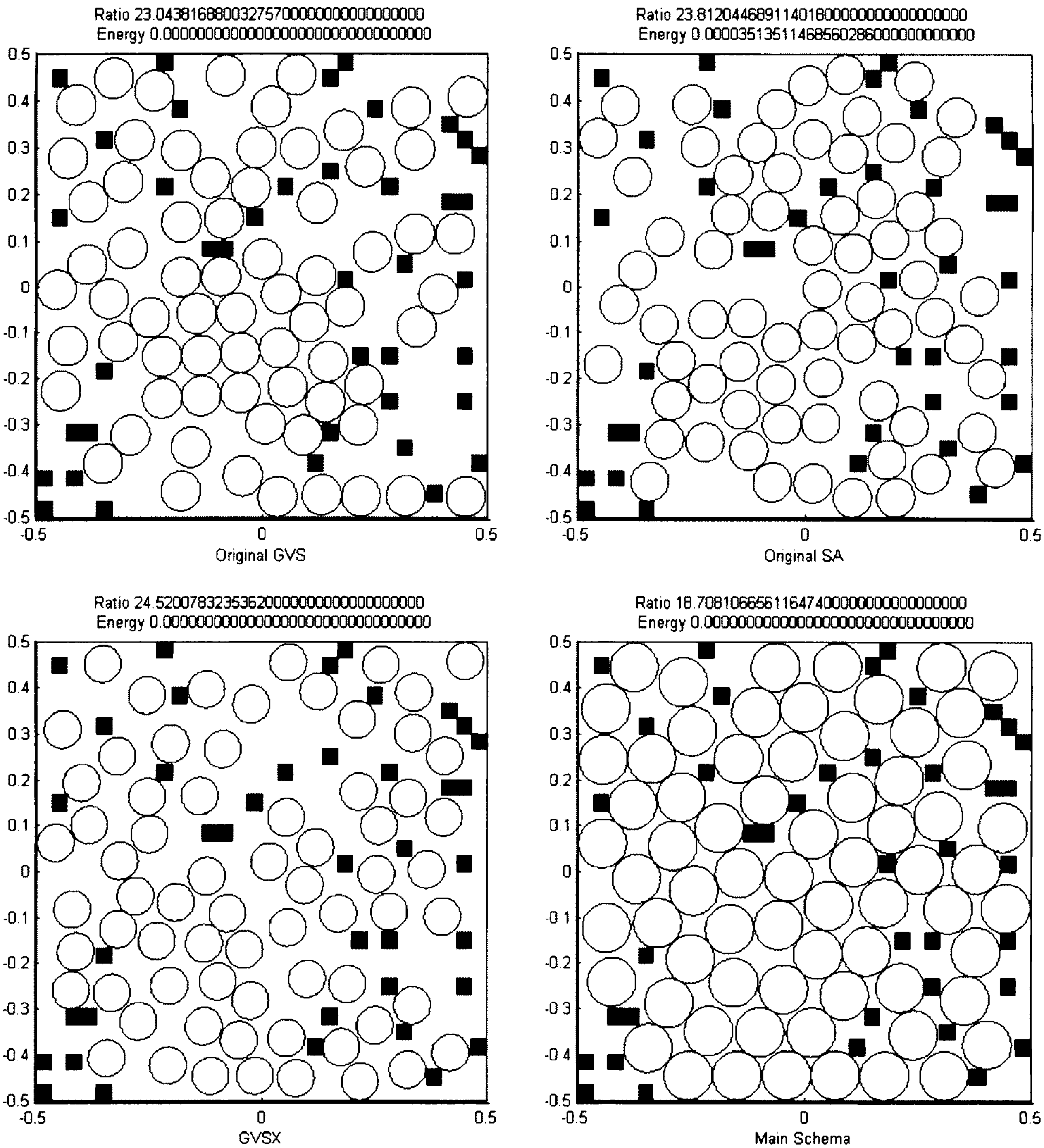


Figure 5.6: Experimental Result: 70 circle packing in a damaged square, $[40/30^2]$.

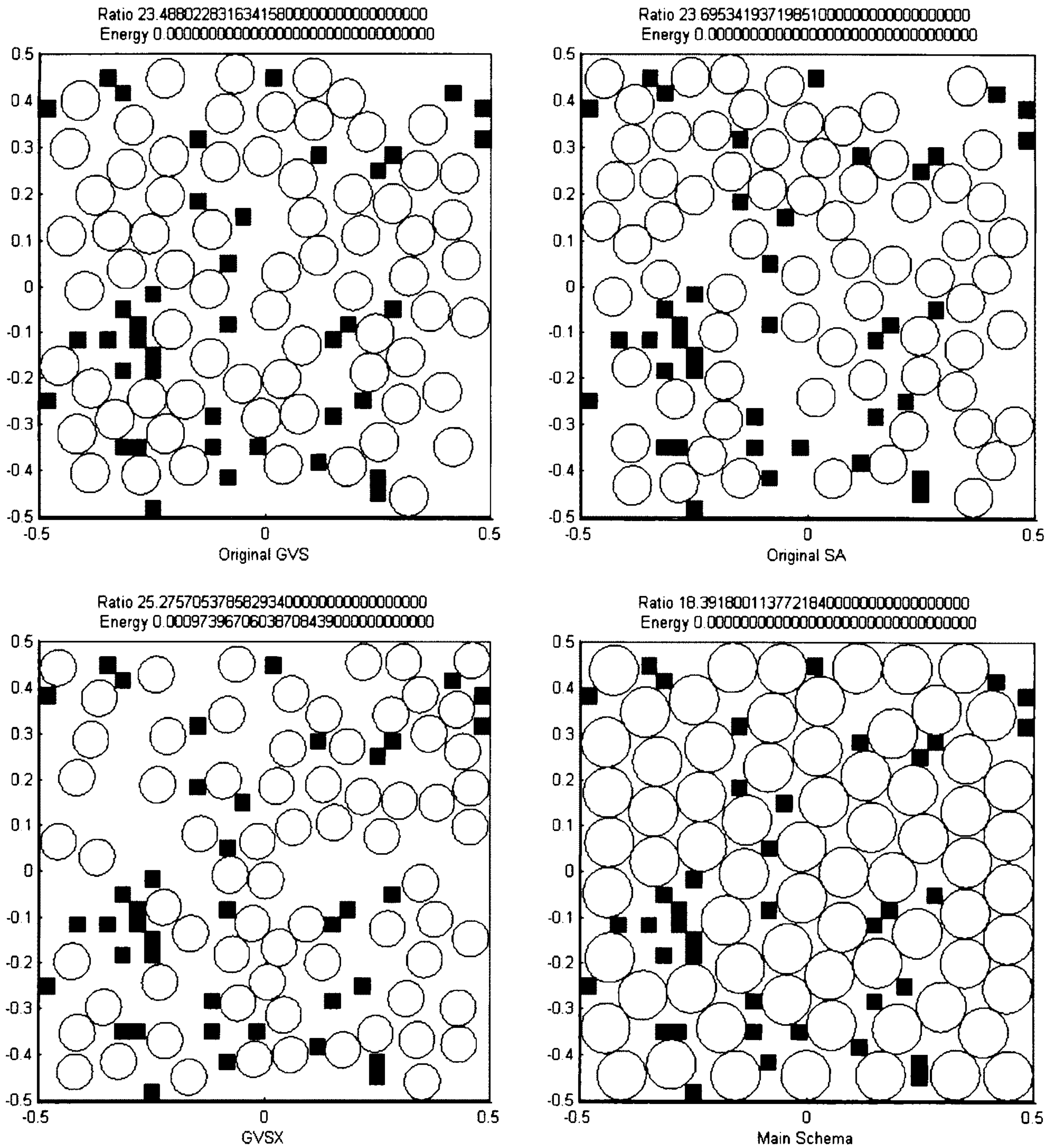
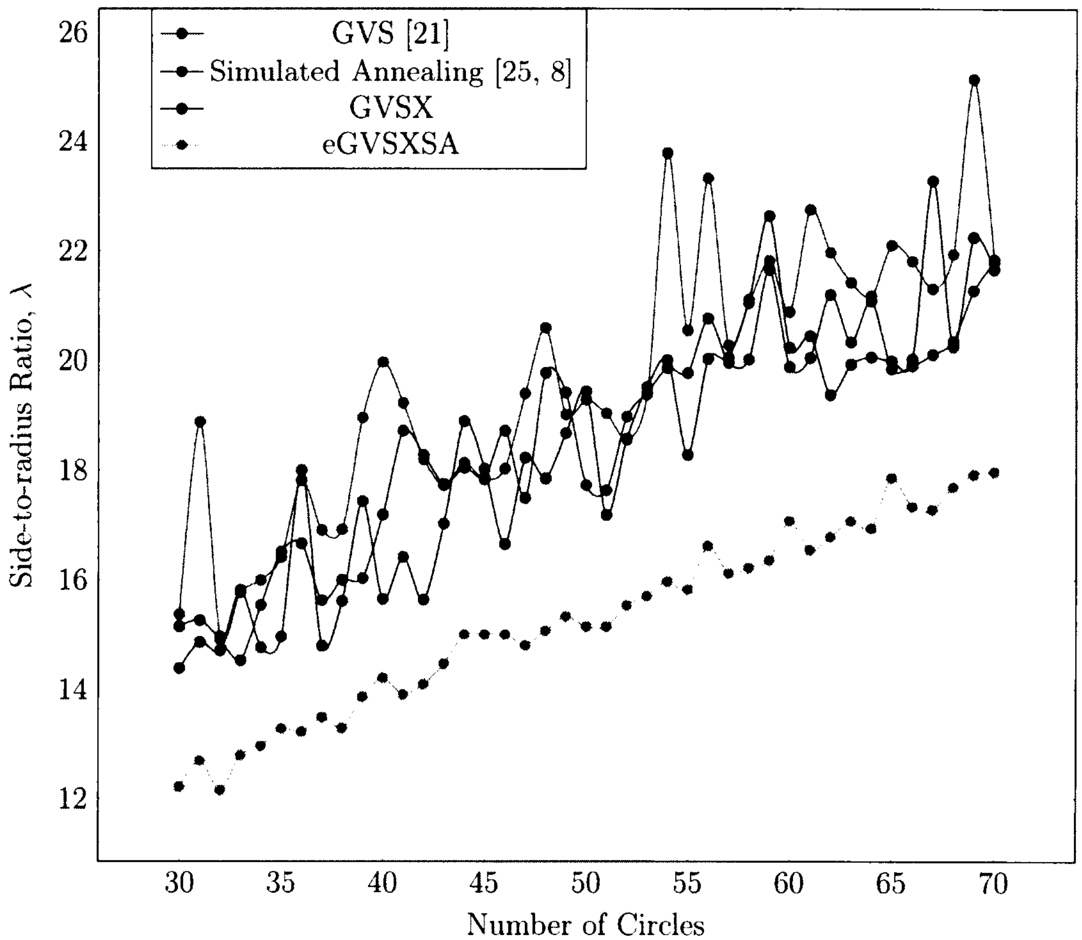


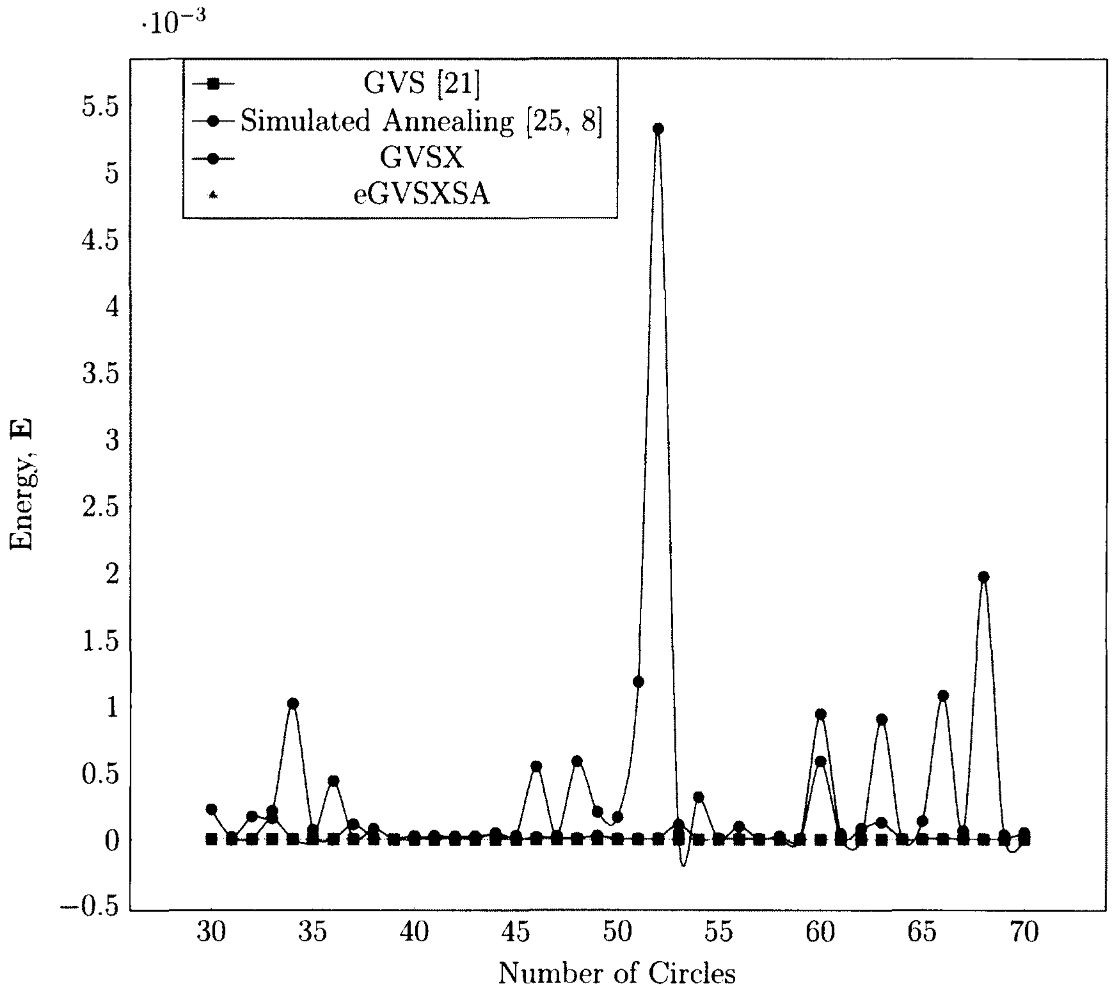
Figure 5.7: Comparison of side-to-radius λ found by the GVS, SA, GVSX and the eGVSXSA Schema on packing 30 to 70 circles in $[20/30^2]$



	Original GVS[21]		Original Simulated Annealing[25, 8]		GVSX		eGVSXSA	
Name	ratio(λ)	energy($\mathbf{E}$)	ratio(λ)	energy($\mathbf{E}$)	ratio(λ)	energy($\mathbf{E}$)	ratio(λ)	energy($\mathbf{E}$)
30	15.3600595286619530	0.0000000000000000	15.1399605925879260	0.0000000000000000	14.3826278646760880	0.0002240120286048	12.2093151957066580	0.0000000000000000
31	18.8670356238221650	0.0000000000000000	15.2549934154604380	0.0000000000000000	14.8581009108464850	0.0000151281121134	12.6885282922690460	0.0000000000000000
32	14.9696484629882750	0.0000000000000000	14.9013305688750620	0.0000000000001541	14.7018982435651980	0.0001707994965878	12.1419674693214910	0.0000000000000000
33	15.7645933469033230	0.0000000000000000	14.5230422987142780	0.0001548770013863	15.8098780197681100	0.0002106945302558	12.7868329042972770	0.0000000000000000
34	15.9942702018679730	0.0000000000000000	15.5314291096676950	0.0000000000000000	14.7609009216372660	0.0010173518730069	12.9549695939498510	0.0000000000000000
35	16.4097978658035540	0.0000000000000000	16.5007250070161680	0.0000000000000000	14.9588207328809930	0.0000692192811028	13.2737793110457730	0.0000000000000000
36	17.8132436114625320	0.0000000000000000	16.6579410035363300	0.0004374548367536	17.9983054824610990	0.0000021038060058	13.2166621666882060	0.0000000000000000
37	16.9011896715064720	0.0000000000000000	15.6226598064113520	0.0000000000000004	14.7881484813926110	0.0001111867128159	13.4738615182292010	0.0000000000000000
38	16.9162168575059230	0.0000000000000000	15.9959717085076320	0.0000792624976757	15.6136429191592430	0.0000087161965128	13.2810603306348810	0.0000000000000000
39	18.9470034253808780	0.0000000000000000	16.0261538110410340	0.0000000000000005	17.4278889487935520	0.0000000000000000	13.8541289009979420	0.0000000000000000
40	19.9608070182522790	0.0000000000000000	17.1850332095822830	0.0000001805099090	15.6489440065234260	0.0000231201062777	14.2012007503842330	0.0000000000000000
41	19.2151778721519580	0.0000000000000000	18.7085033109734550	0.0000249257384063	16.4148758832177780	0.0000000000000000	13.9012950678297320	0.0000000000000000
42	18.1941661579948860	0.0000000000000000	18.2761368287915720	0.0000169116199289	15.6321237237956830	0.0000205269003421	14.0863800242564100	0.0000000000000000
43	17.7227944524425740	0.0000000000000000	17.7478541462638050	0.0000233723754611	17.0181743695146320	0.0000005575159553	14.4614975305793830	0.0000000000000000
44	18.1250235526118040	0.0000000000000000	18.0368102026227890	0.0000336132245724	18.8824004531564430	0.0000461266018298	15.0000000026352570	0.0000000000000000
45	17.8262453053923690	0.0000000000000000	17.8657474197645140	0.0000267224046387	18.0171850093788780	0.0000006701217764	14.9999999983601140	0.0000000000000000
46	18.0198173300403540	0.0000000000000000	18.7102432057285810	0.0005458122081565	16.6464281575256800	0.0000140465280713	15.0000000138566630	0.0000000000000000
47	19.3888798663019200	0.0000000000000000	17.4882450895097110	0.0000234538172212	18.2190029945049550	0.0000098346567999	14.7992081433500540	0.0000000000000000
48	20.5894727185166420	0.0000000000000000	19.7626240812427980	0.0005843859859434	17.8476020318845060	0.0000068722649220	15.0667479861408890	0.0000000000000000
49	19.0043463770487120	0.0000000000000000	19.4057233819916700	0.0002048795889991	18.6652661969253730	0.0000256952201961	15.3278537971023600	0.0000000000000000
50	19.2711312048271490	0.0000000000000000	17.7303040250089300	0.0001654228229829	19.4268786101901650	0.0000038362150174	15.1427729388373140	0.0000000000000000
51	19.0284455848120330	0.0000000000000000	17.6312918397617440	0.0011816468849379	17.1791025325968580	0.0000038022455152	15.1392713211858500	0.0000000000000000
52	18.5565478814627270	0.0000000000000000	18.9687760278743550	0.0053207440326906	18.5675419781556920	0.0000055917118162	15.5359600790415190	0.0000000000000000
53	19.3709236859500140	0.0000000000000000	19.3919567237183940	0.0000434253101611	19.5089902749047680	0.0001096174132748	15.7058796373065230	0.0000000000000000
54	23.7742963901871250	0.0000000000000000	19.8589942846831780	0.0003168520991790	19.9999996042115350	0.0000000000000985	15.9702247006575390	0.0000000000000000
55	20.5402758791071460	0.0000000000000000	19.7633858855938730	0.0000000000000076	18.2752521933491000	0.0000041100904624	15.8223173750945740	0.0000000000000000
56	23.3217176420905050	0.0000000000000000	20.7614695225339040	0.0000938260910861	20.0227822646496630	0.0000015225708013	16.6225689758416910	0.0000000000000000
57	20.2711818824910100	0.0000000000000000	19.9508310148817360	0.0000034437490742	20.0433272584658580	0.0000028290737546	16.1192520723505550	0.0000000000000000
58	21.0420944841323350	0.0000000000000000	20.0117939428323980	0.0000207358545542	21.1017121019110830	0.0000007738487568	16.2184043091221780	0.0000000000000000
59	21.8157155518828710	0.0000000000000000	21.6477380656723350	0.0000000000000000	22.6331169181481120	0.0000038267806840	16.3543161150830120	0.0000000000000000
60	20.8791313298842810	0.0000000000000000	19.8726721992353180	0.0005858754169480	20.2263083927024140	0.0009435256269657	17.0710770476096410	0.0000000000000000
61	22.7450835034601940	0.0000000000000000	20.0406019393348630	0.0000046853140467	20.4398675620724430	0.0000406298071780	16.5437244928949170	0.0000000000000000
62	21.9664306457316840	0.0000000000000000	21.1965611539959160	0.0000000000000000	19.3706568822843440	0.0000800464749259	16.7838241775306220	0.0000000000000000
63	21.4128503812925570	0.0000000000000000	20.3288906501482740	0.0009038728601756	19.9235831532264600	0.0001257498398068	17.0720126328351040	0.0000000000000000
64	21.1689575763450190	0.0000000000000000	21.0774314442015510	0.0000000000000000	20.0524199293439980	0.0000020493487108	16.9396027888065430	0.0000000000000000
65	22.1016766234400530	0.0000000000000000	19.8387286772959040	0.0001349402888516	19.9825260006059580	0.0000034441865068	17.8590193117760380	0.0000000000000000
66	21.8021933250530400	0.0000000000000000	19.8985855189225380	0.0010816923798531	20.0187590935122160	0.0000053706602399	17.3346681558371710	0.0000000000000000
67	21.3016008834996310	0.0000000000000000	20.1000089458578660	0.0000604551098218	23.2751147588678510	0.0000017158736915	17.2802025151309380	0.0000000000000000
68	21.9337352319386800	0.0000000000000000	20.3414660527200230	0.0019751265880884	20.2456101682339380	0.0000028678641098	17.6925026111845440	0.0000000000000000
69	25.1192734420609830	0.0000000000000000	21.2660506545390430	0.0000316018831713	22.2436619963907770	0.0000023116616007	17.9157166266140210	0.0000000000000000
70	21.8295378399979820	0.0000000000000000	21.7815871311081040	0.0000062774038483	21.6465699563223030	0.0000468996040701	17.9645197599917270	0.0000000000000000

Table 5.2: Statistics of comparing GVS, SA, GVSX and eGVSXSA Schema on packing 30 to 70 circles, [20/30^2]

Figure 5.8: Comparison of energy **E** found by GVS, SA, GVSX and the eGVSXSA on packing 30 to 70 circles in $[20/30^2]$



	Original GVS	Original SA	GVSX	eGVSXSA
Name	hour(s)	hour(s)	hour(s)	hour(s)
30	2.0000	1.4165	1.7394	14.0637
31	2.0000	1.4283	1.6459	14.1356
32	2.0000	1.4474	2.4097	14.1266
33	2.0000	1.4934	0.8685	14.2185
34	2.0000	1.4145	1.8828	14.2212
35	2.0000	1.4224	1.5015	14.3389
36	2.0000	1.4294	0.6387	14.3233
37	2.0000	1.4339	2.2288	14.3469
38	2.0000	1.4487	2.4358	14.2483
39	2.0000	1.4497	1.6351	14.3095
40	2.0000	1.4645	1.8326	14.3296
41	2.0000	1.4383	1.6822	14.4491
42	2.0000	1.4705	0.1409	14.8928
43	2.0000	1.4158	2.1513	15.1305
44	2.0000	1.4445	2.2258	15.4049
45	2.0000	1.4801	1.8268	15.5403
46	2.0000	1.4581	1.6444	15.5308
47	2.0000	1.4196	1.7586	15.5308
48	2.0000	1.4891	1.3118	15.5259
49	2.0000	1.4194	1.8303	16.5543
50	2.0000	1.4572	2.0021	16.5825
51	2.0000	1.4569	2.9529	16.5403
52	2.0000	1.4231	2.5265	16.5308
53	2.0000	1.4852	2.0318	16.5259
54	2.0000	1.4354	2.1203	16.5543
55	2.0000	1.4948	1.5365	16.5825
56	2.0000	1.4385	1.8649	17.3924
57	2.0000	1.4754	1.9031	17.9593
58	2.0000	1.4992	1.5663	17.0502
59	2.0000	1.4789	1.8976	17.4033
60	2.0000	1.4518	2.6554	17.6073
61	2.0000	1.4774	2.7225	17.8174
62	2.0000	1.4655	1.7768	17.1332
63	2.0000	1.4854	1.7445	17.0832
64	2.0000	1.4057	2.7633	17.8319
65	2.0000	1.4235	2.0689	17.9612
66	2.0000	1.4513	1.6117	17.8262
67	2.0000	1.4975	2.7734	17.3774
68	2.0000	1.4708	2.0806	17.0377
69	2.0000	1.4068	2.4094	17.5065
70	2.0000	1.4853	2.3352	17.4252

Table 5.3: Computational time by GVS, SA, GVSX and eGVSXSA on packing 30 to 70 circles, $[20/30^2]$

Chapter 6

Summary

In this thesis, I have introduced a variation of the problem of Packing Equal Circles in a Square (PECS) in which the interior of the container may be damaged; the damages are represented by identical square shape objects. I refer to this generalized version of PECS as *Packing Equal Circles in a Damaged Square (PECDS)*.

I have introduced a new heuristic algorithm called *Enhanced Greedy Vacancy Search optimised by Simulated Annealing (eGVSXSA)* for PECDS. The new algorithm iterates an enhanced version of Greedy Vacancy Search algorithm followed by a modified Simulated Annealing algorithm, until the termination condition is met.

I performed a number of experiments to demonstrate the significant advantages of eGVSXSA over the original GVS and SA. The experimental results presented in Chapter 5, indicate a robust performance of eGVSXSA (Figure 5.7).

For future work, we may consider using the eGVSXSA to solve different shapes of damage container such as circle, triangle or rectangular container, etc.

Appendices

Figure 1: Experimental Result: 30 circle packing in a damaged square, $[20/30^2]$.

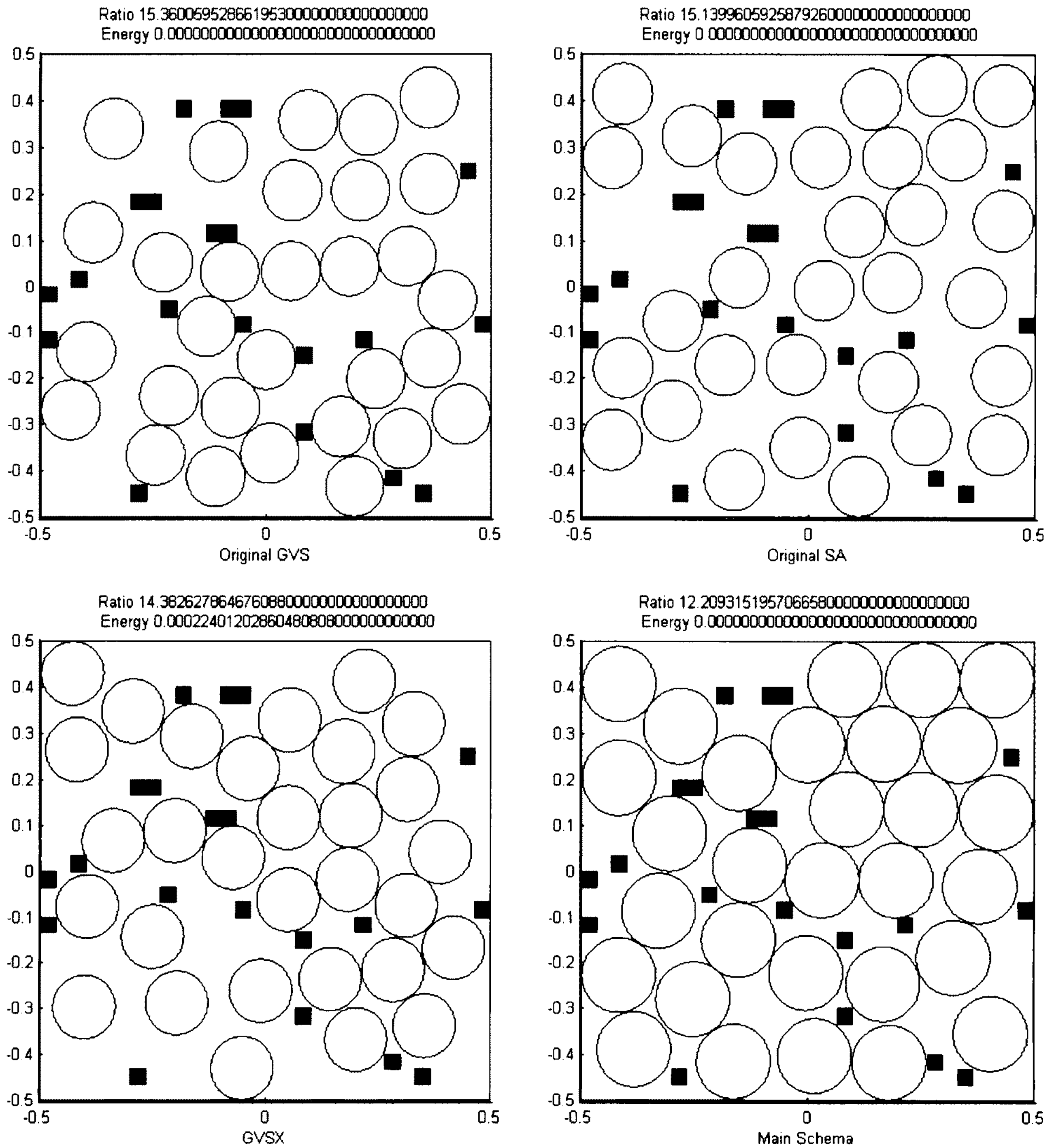


Figure 2: Experimental Result: 31 circle packing in a damaged square, $[20/30^2]$.

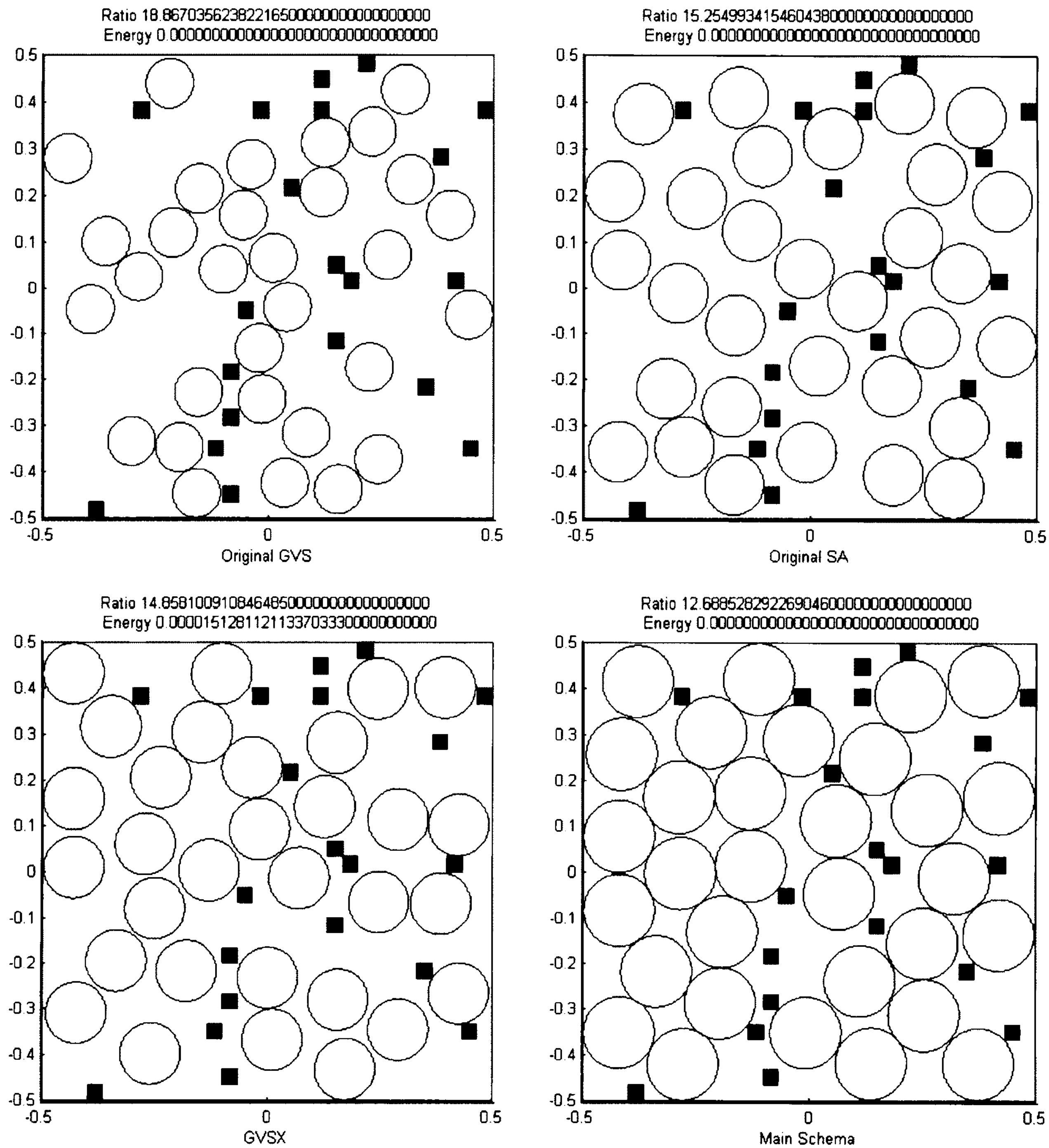


Figure 3: Experimental Result: 32 circle packing in a damaged square, [20/30^2].

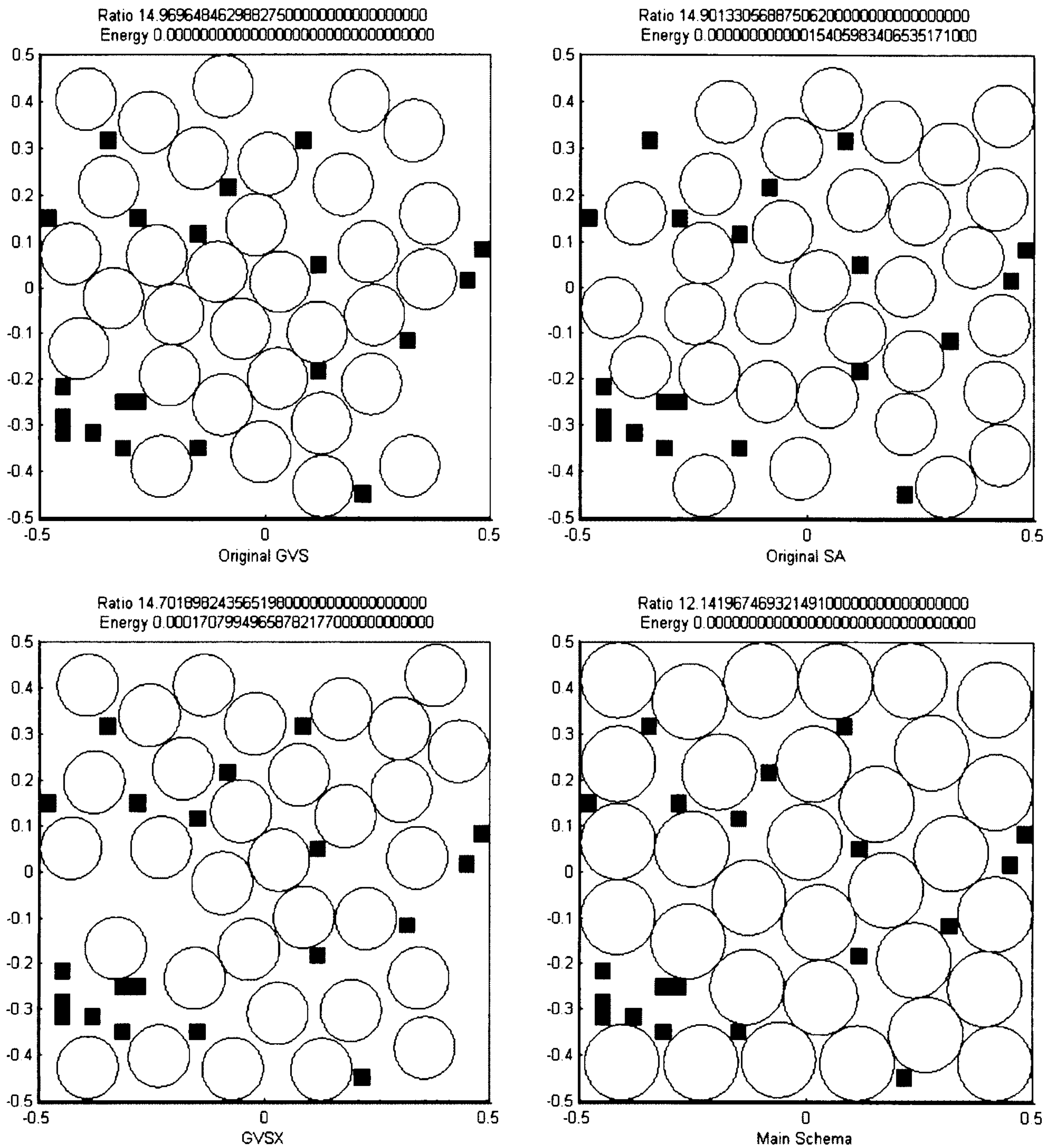


Figure 4: Experimental Result: 33 circle packing in a damaged square, [20/30^2].

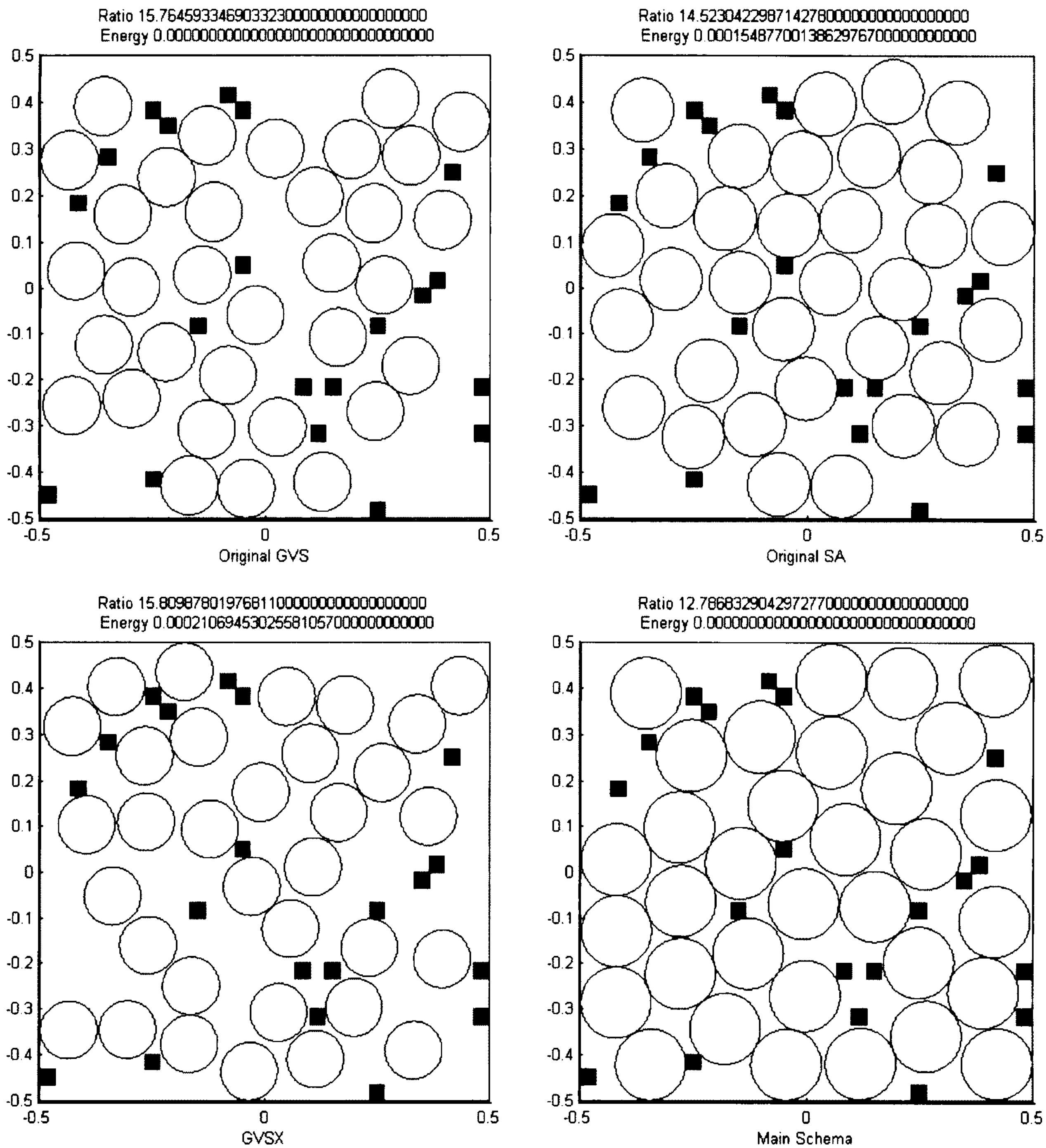


Figure 5: Experimental Result: 34 circle packing in a damaged square, $[20/30^2]$.

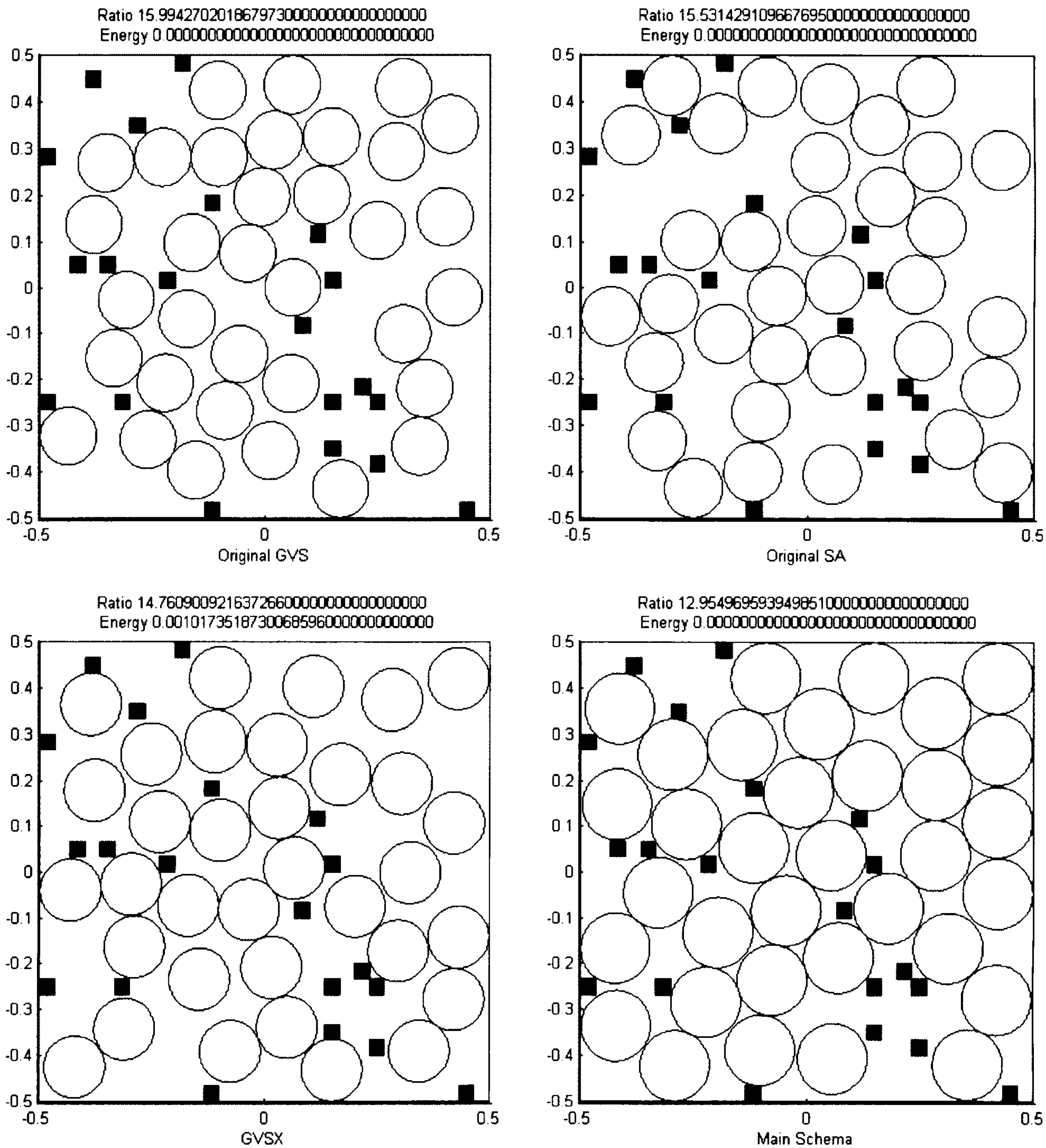


Figure 6: Experimental Result: 35 circle packing in a damaged square, [20/30²].

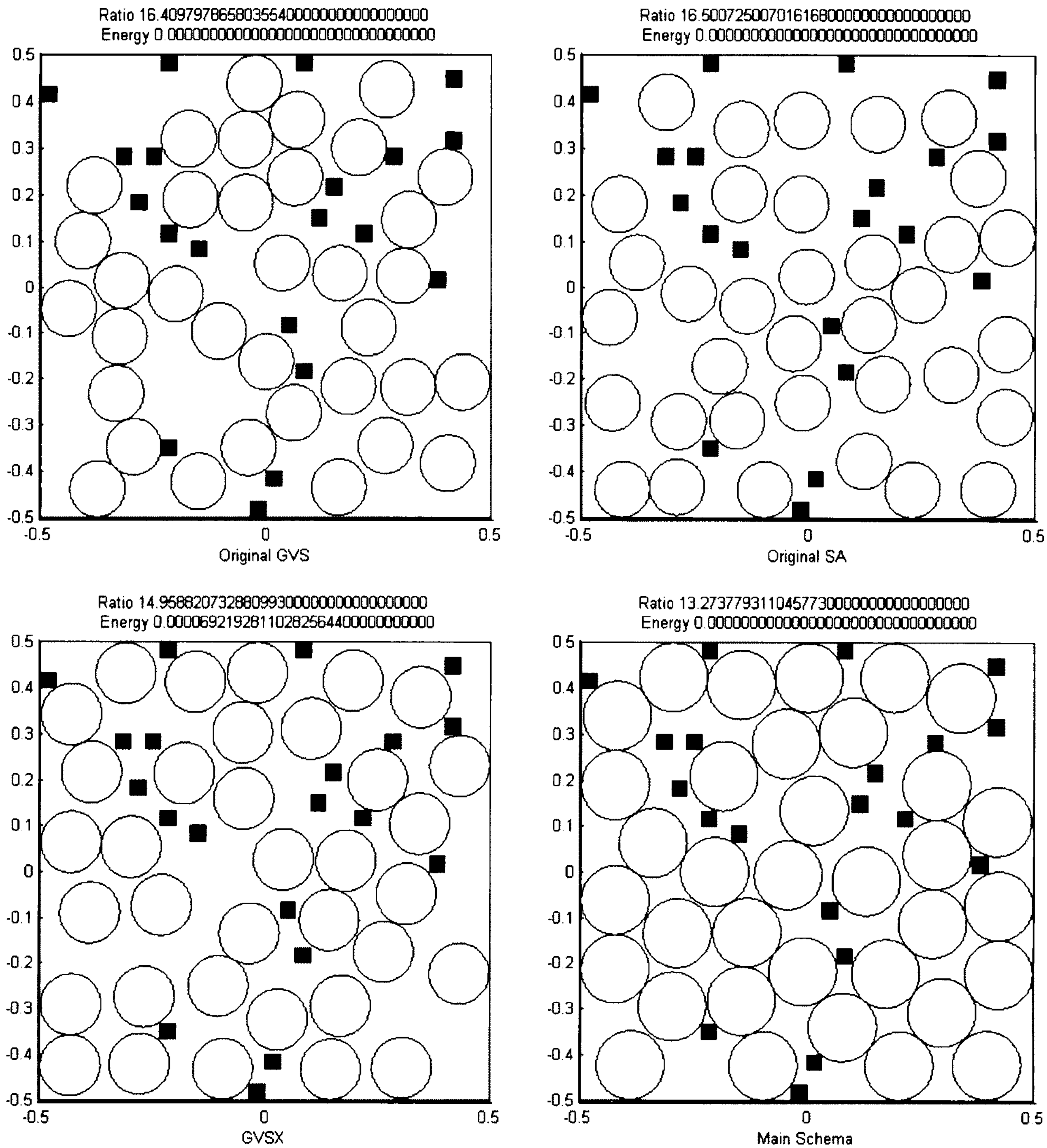


Figure 7: Experimental Result: 36 circle packing in a damaged square, [20/30²].

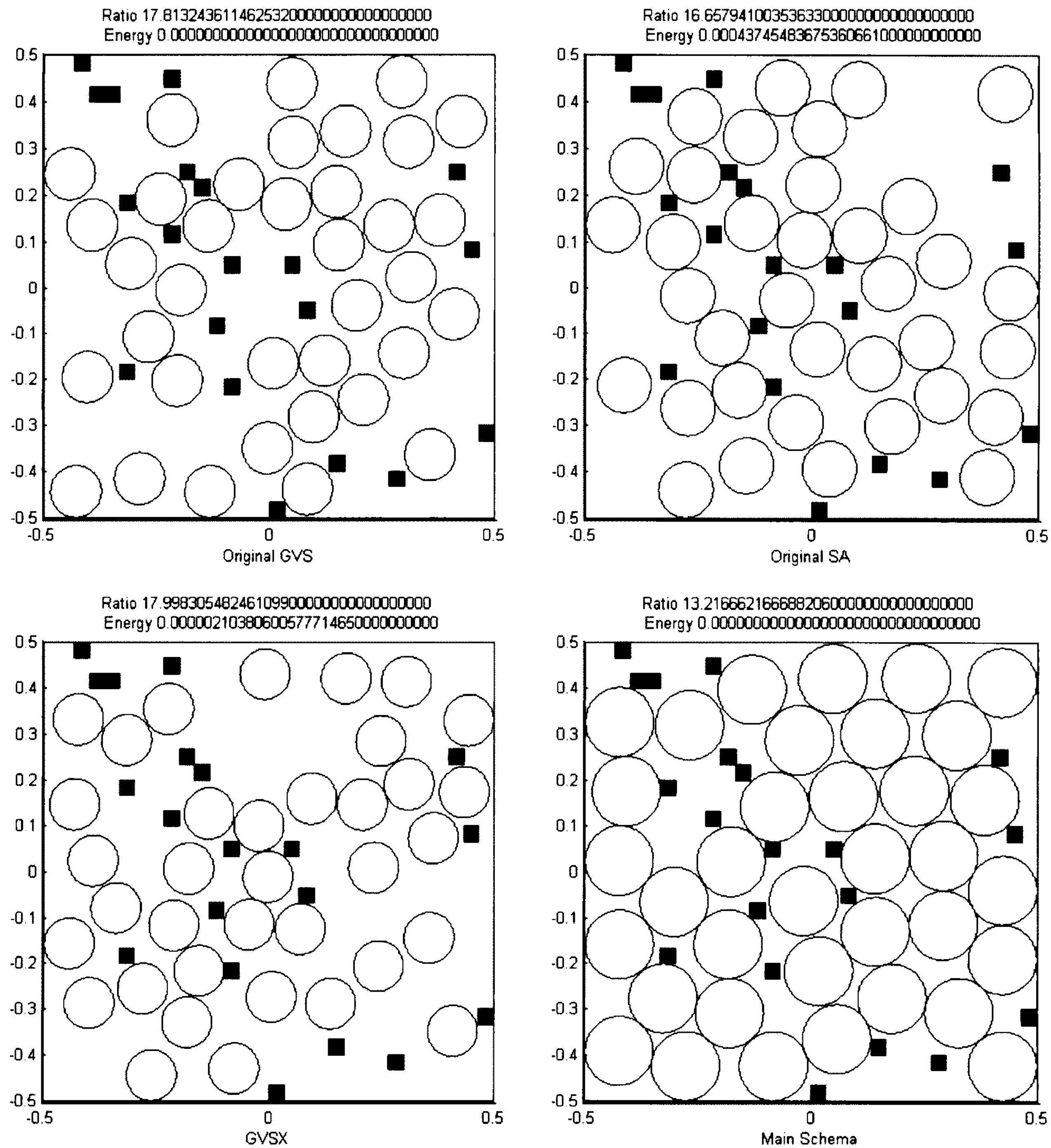


Figure 8: Experimental Result: 37 circle packing in a damaged square, $[20/30^2]$.

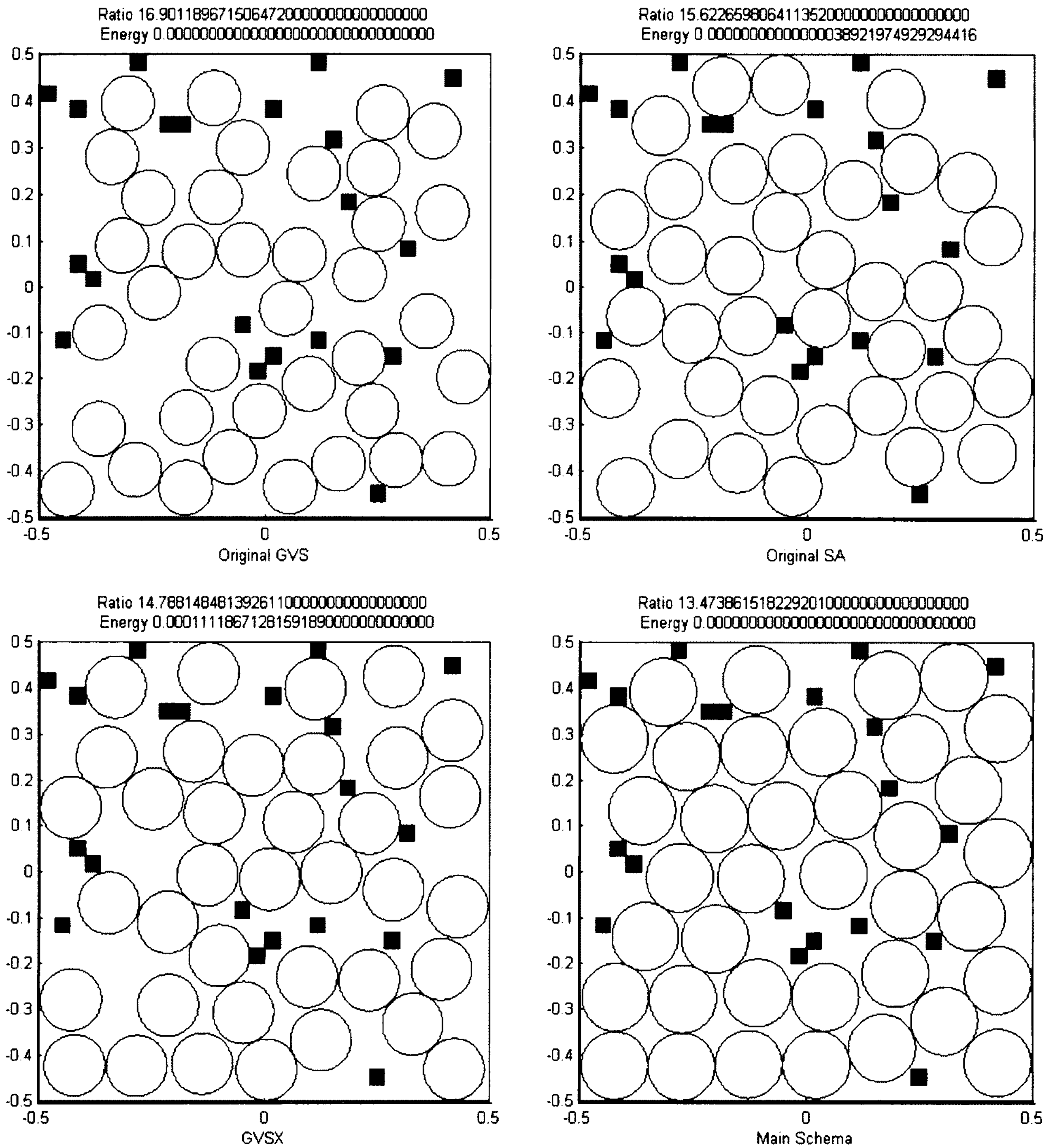


Figure 9: Experimental Result: 38 circle packing in a damaged square, $[20/30^2]$.

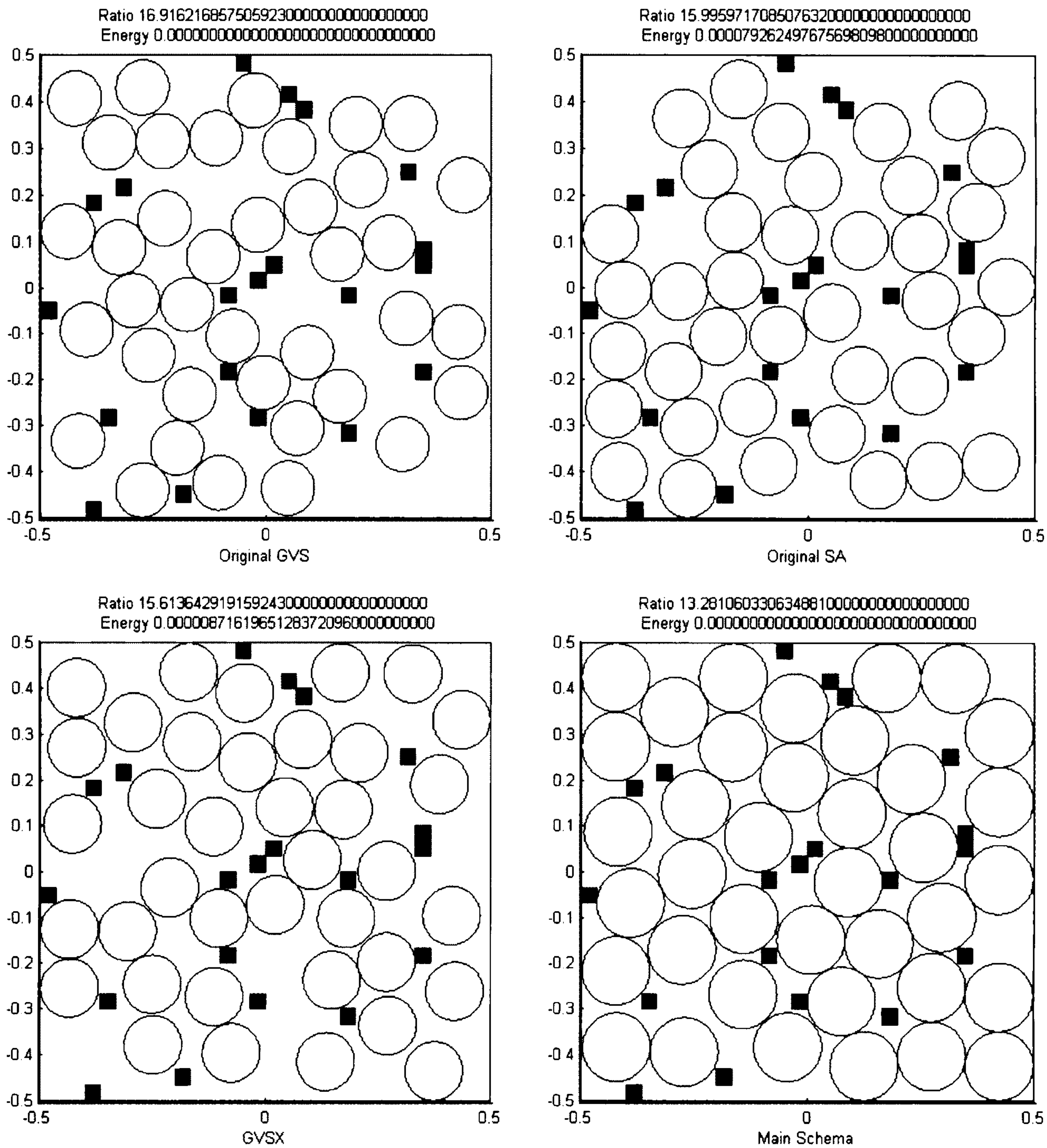


Figure 10: Experimental Result: 39 circle packing in a damaged square, $[20/30^2]$.

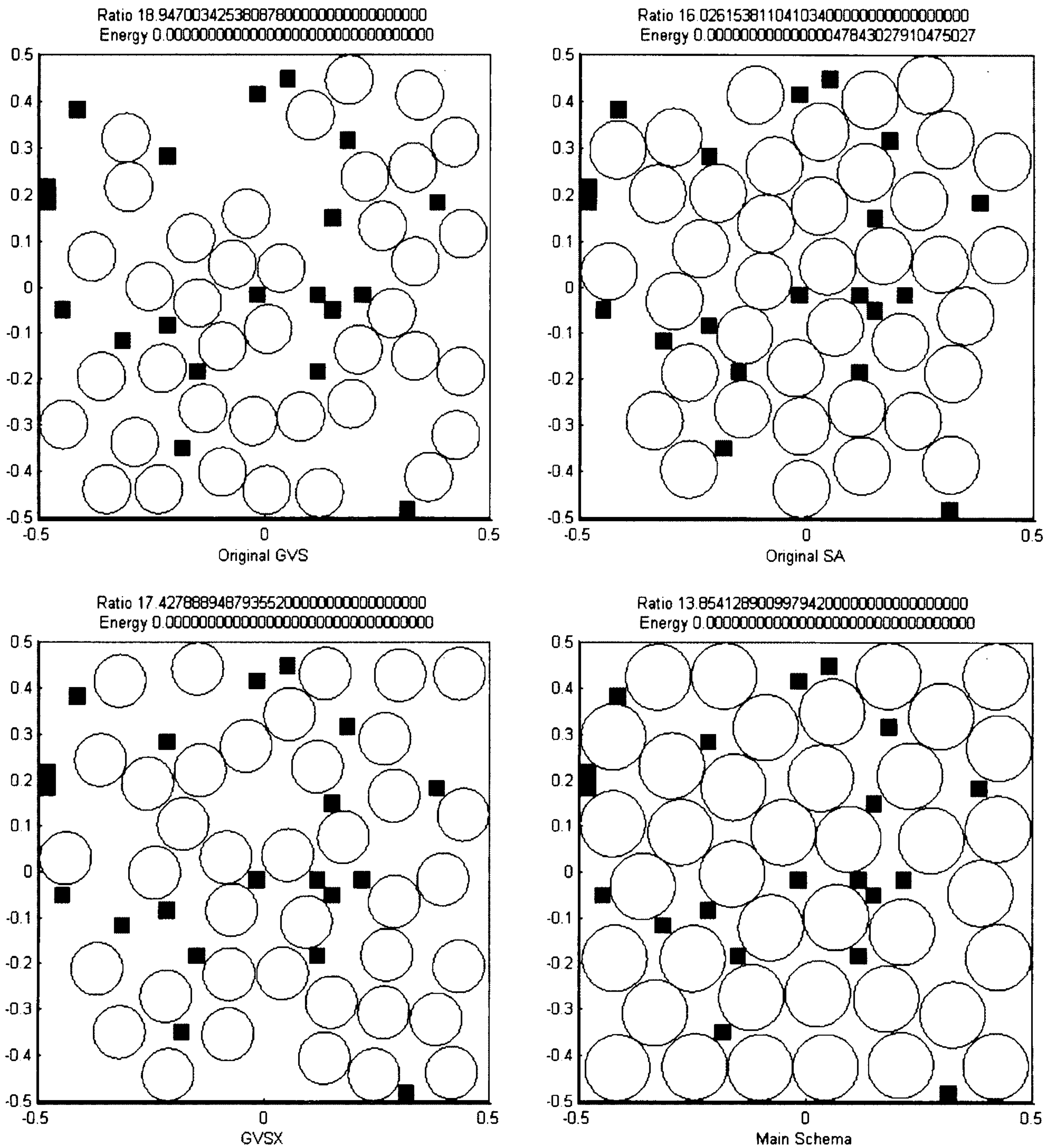


Figure 11: Experimental Result: 40 circle packing in a damaged square, [20/30[2]].

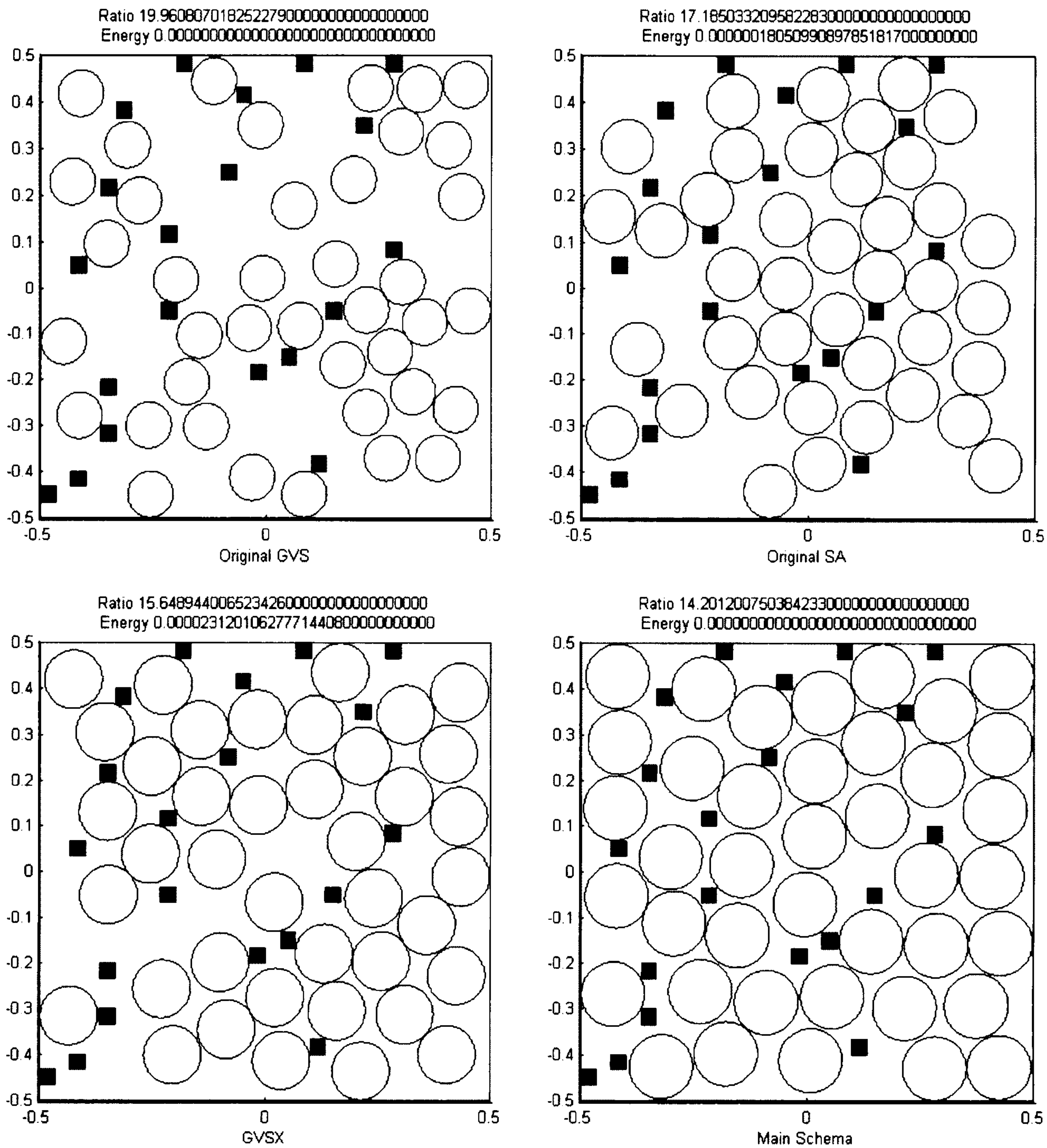


Figure 12: Experimental Result: 41 circle packing in a damaged square, $[20/30^2]$.

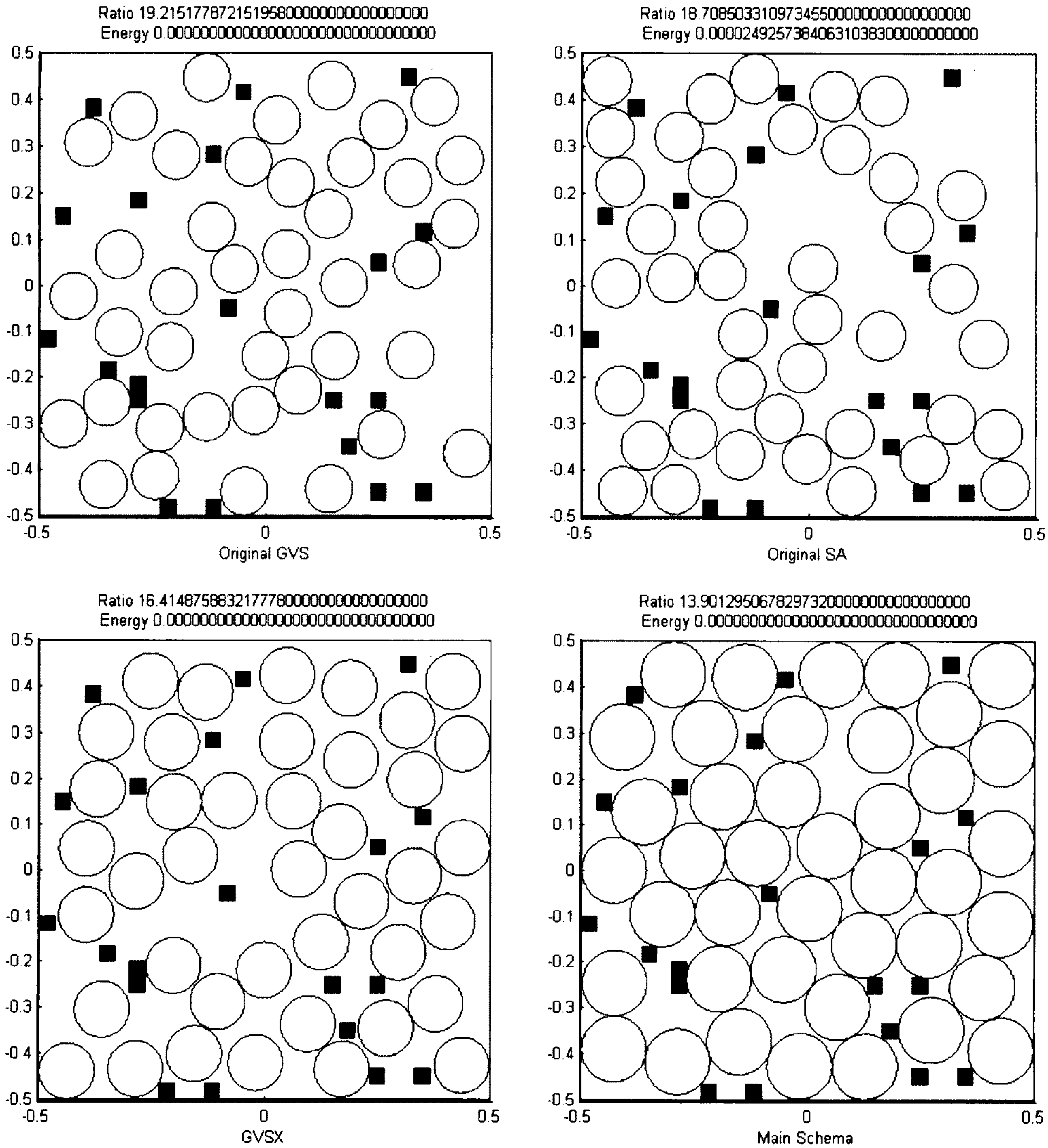


Figure 13: Experimental Result: 42 circle packing in a damaged square, $[20/30^2]$.

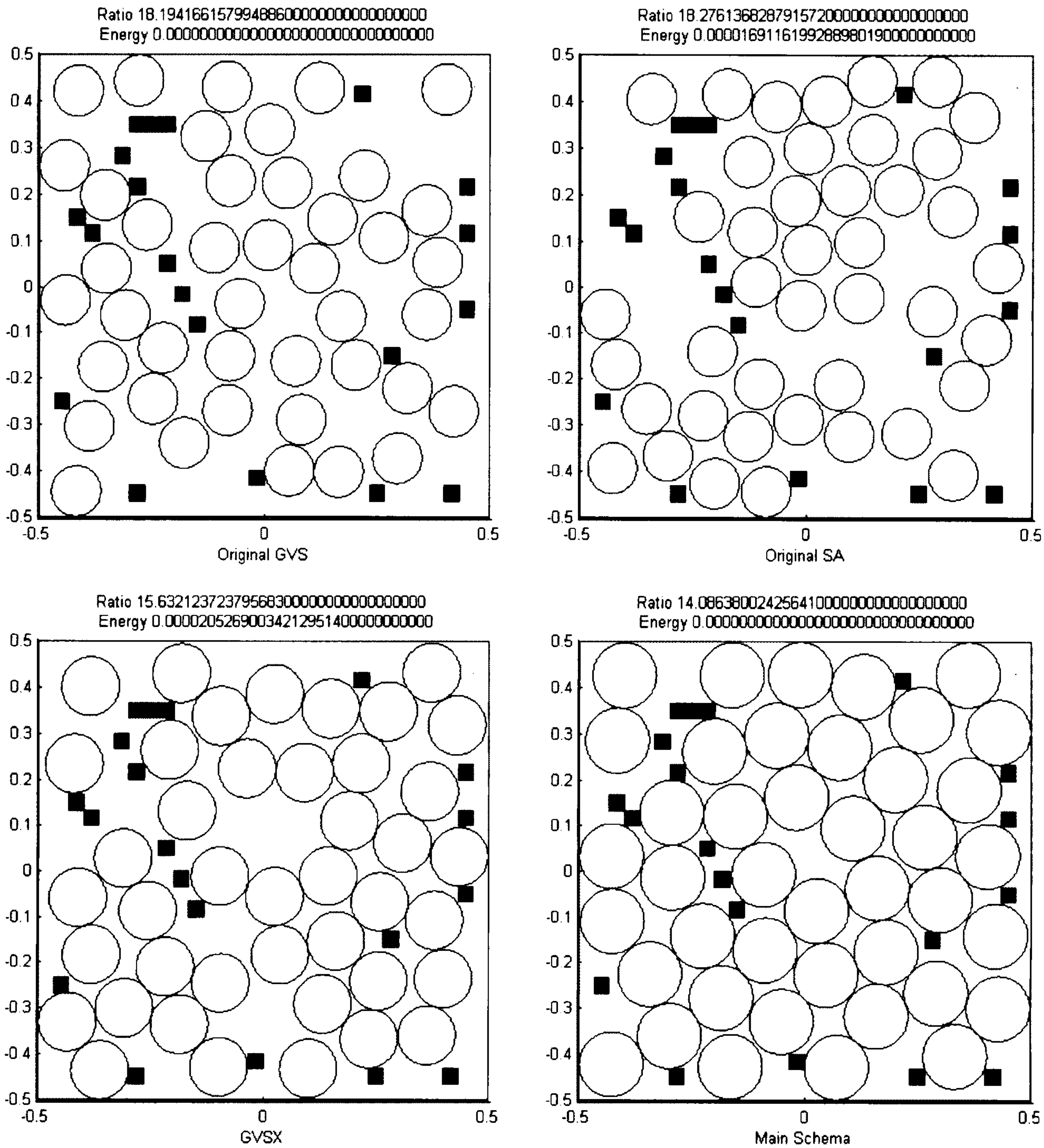


Figure 14: Experimental Result: 43 circle packing in a damaged square, $[20/30^2]$.

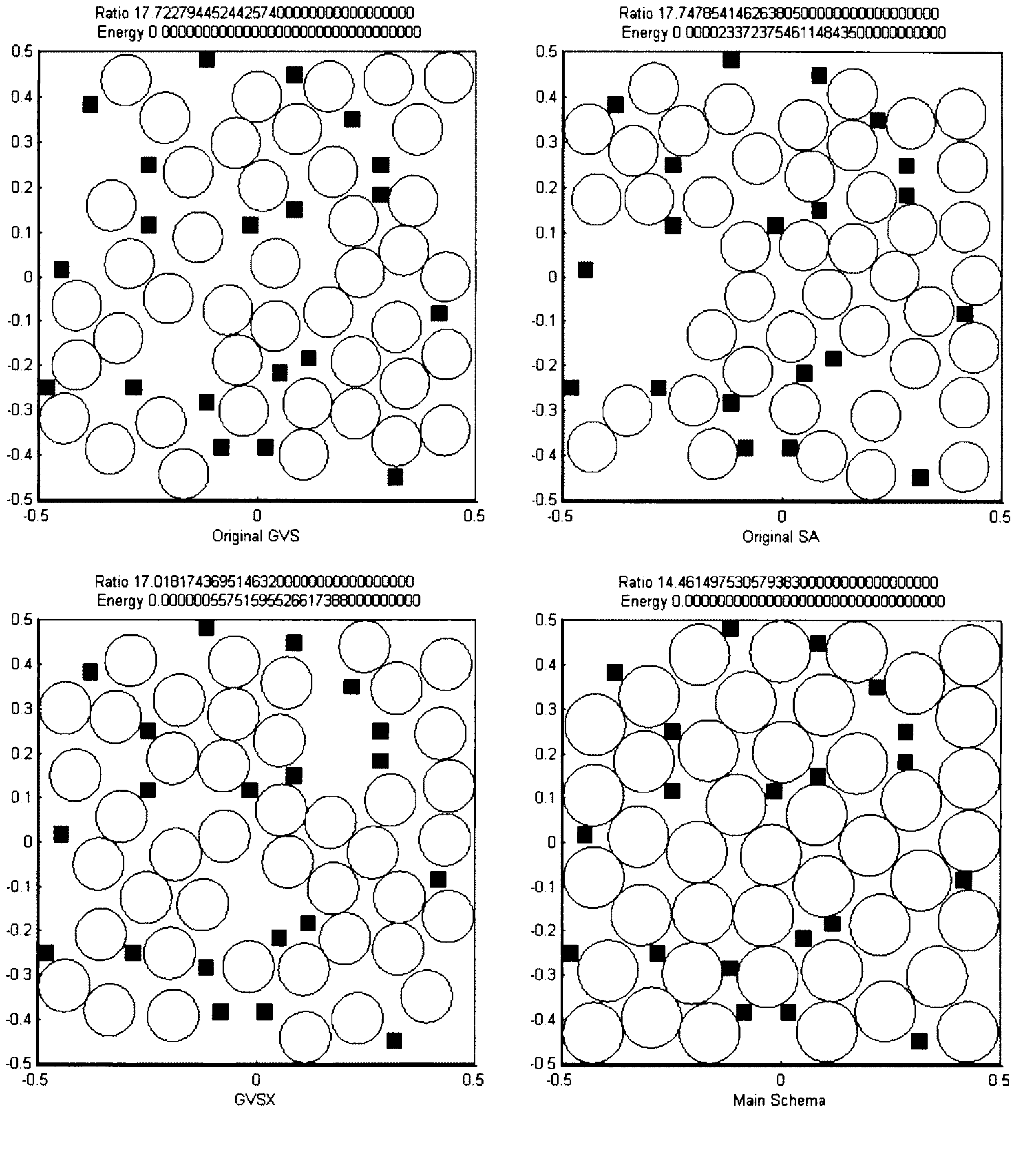


Figure 15: Experimental Result: 44 circle packing in a damaged square, $[20/30^2]$.

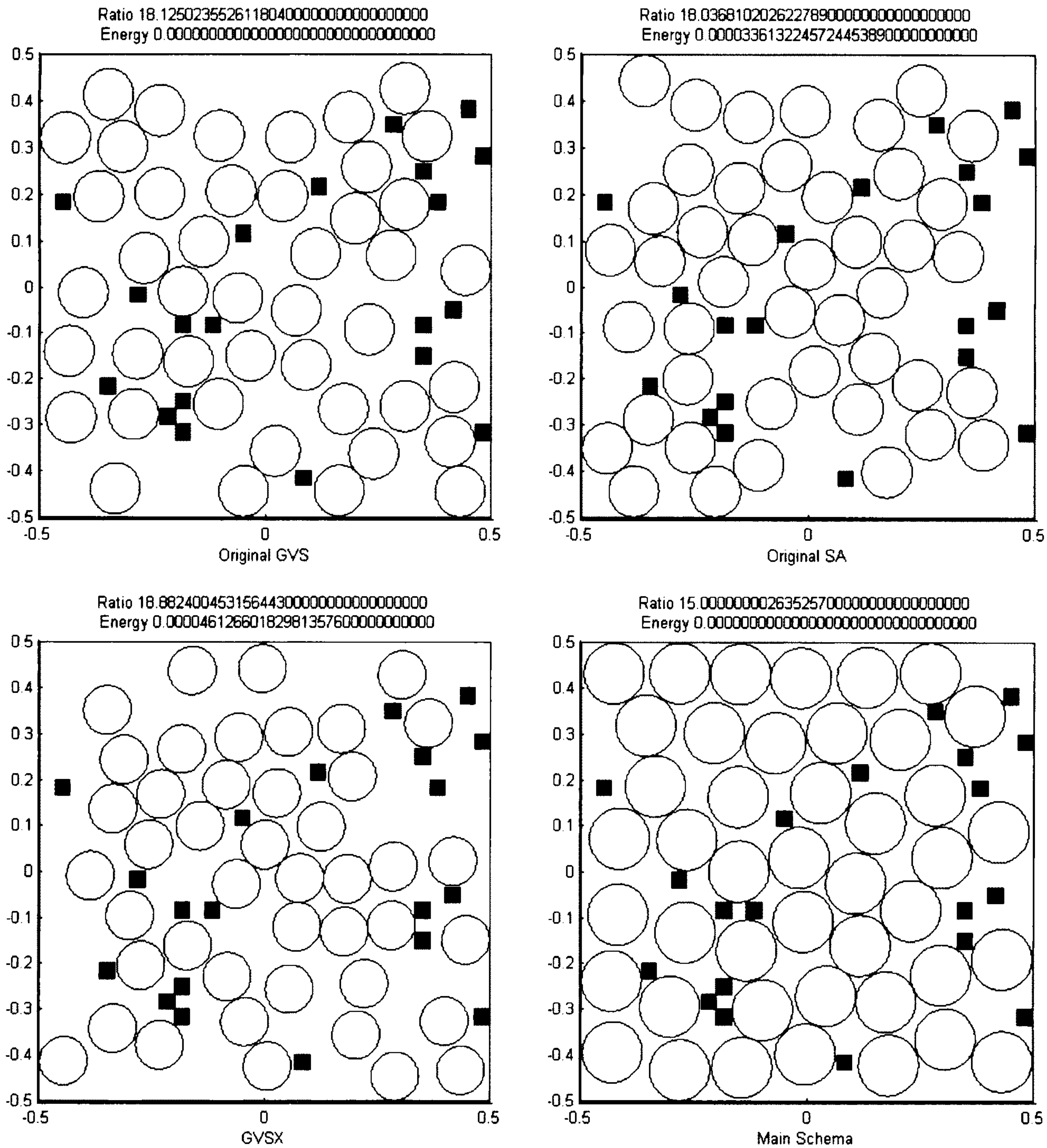


Figure 16: Experimental Result: 45 circle packing in a damaged square, $[20/30^2]$.

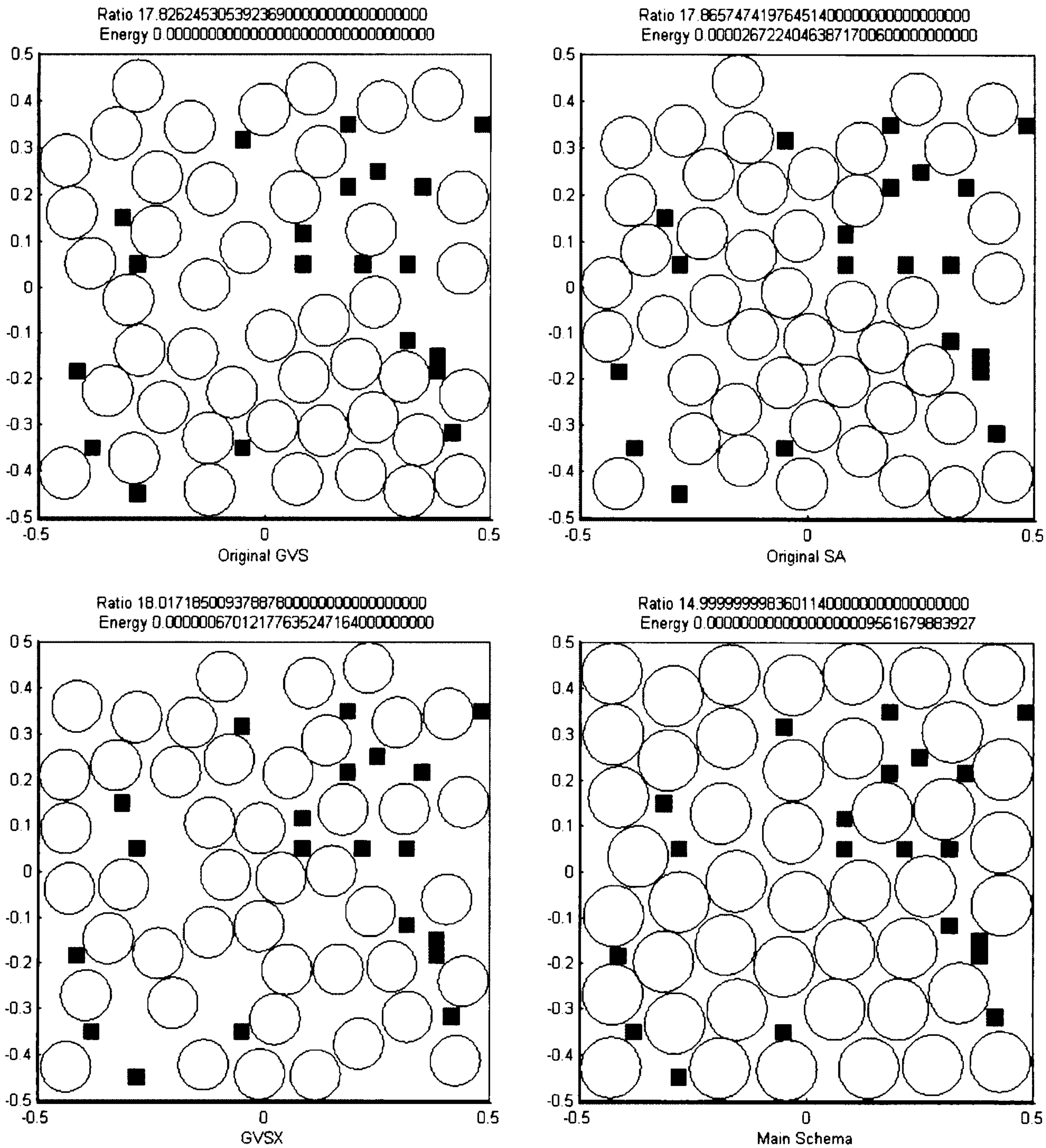


Figure 17: Experimental Result: 46 circle packing in a damaged square, [20/30^2].

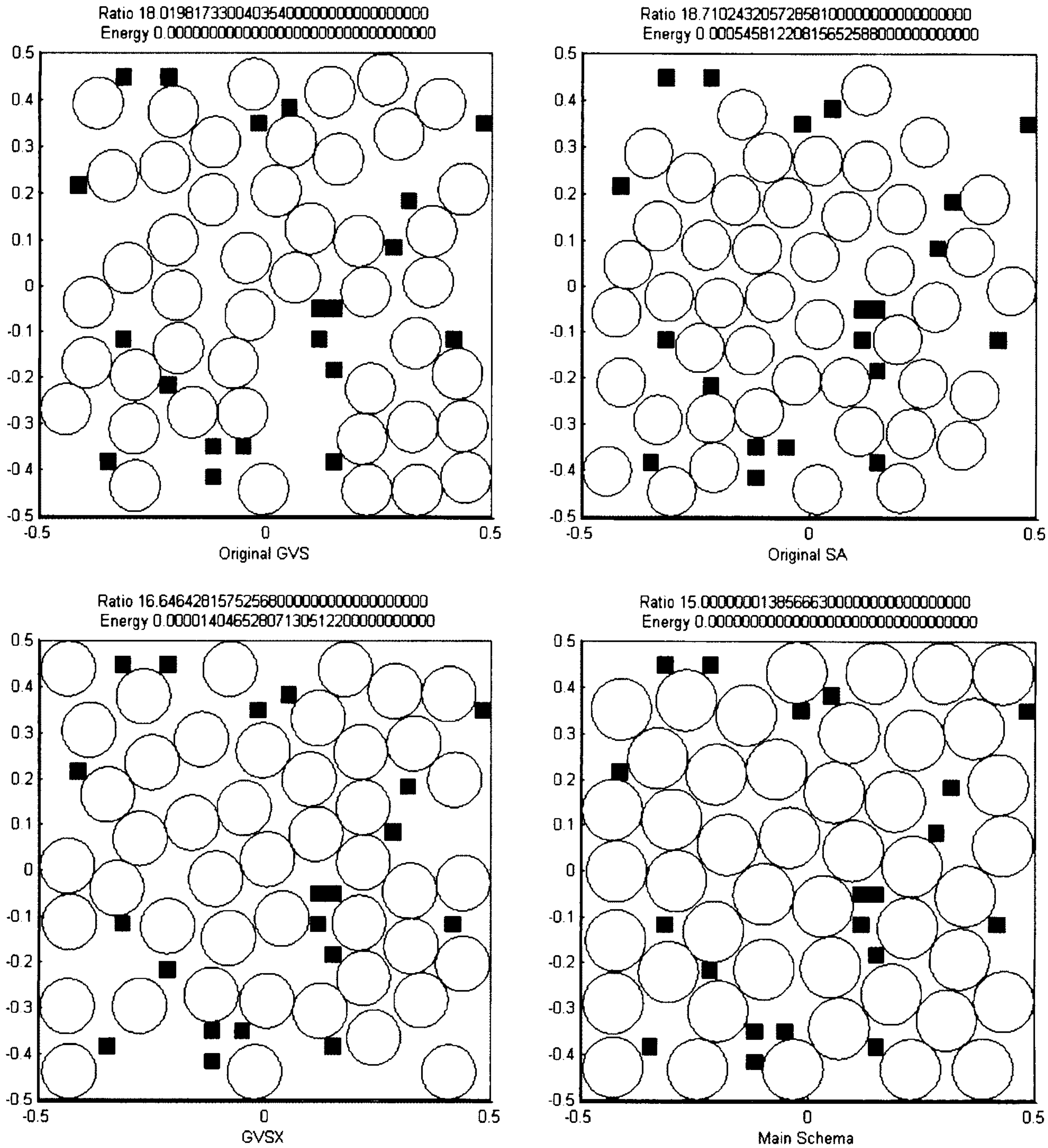


Figure 18: Experimental Result: 47 circle packing in a damaged square, [20/30[2]].

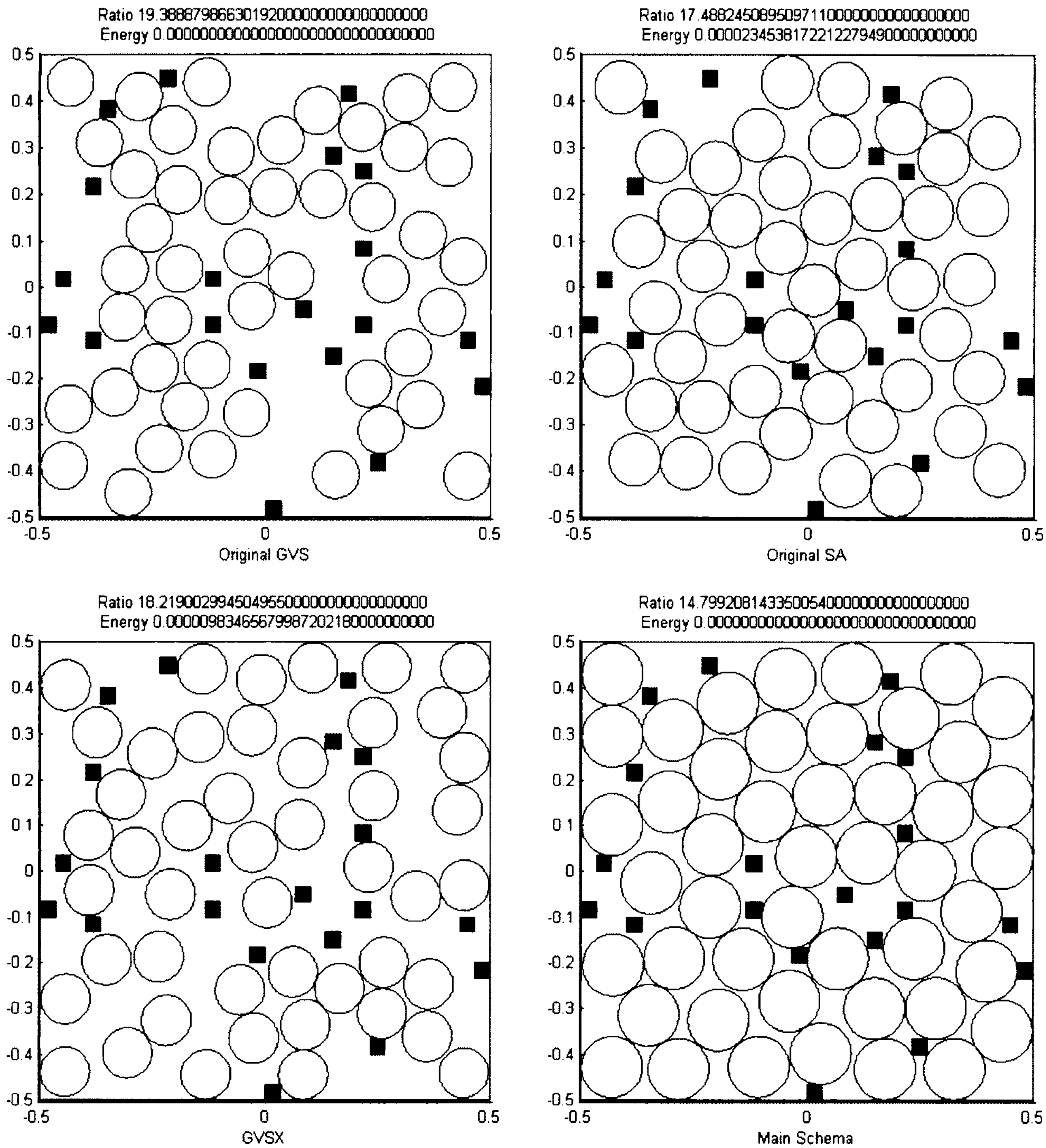


Figure 19: Experimental Result: 48 circle packing in a damaged square, $[20/30^2]$.

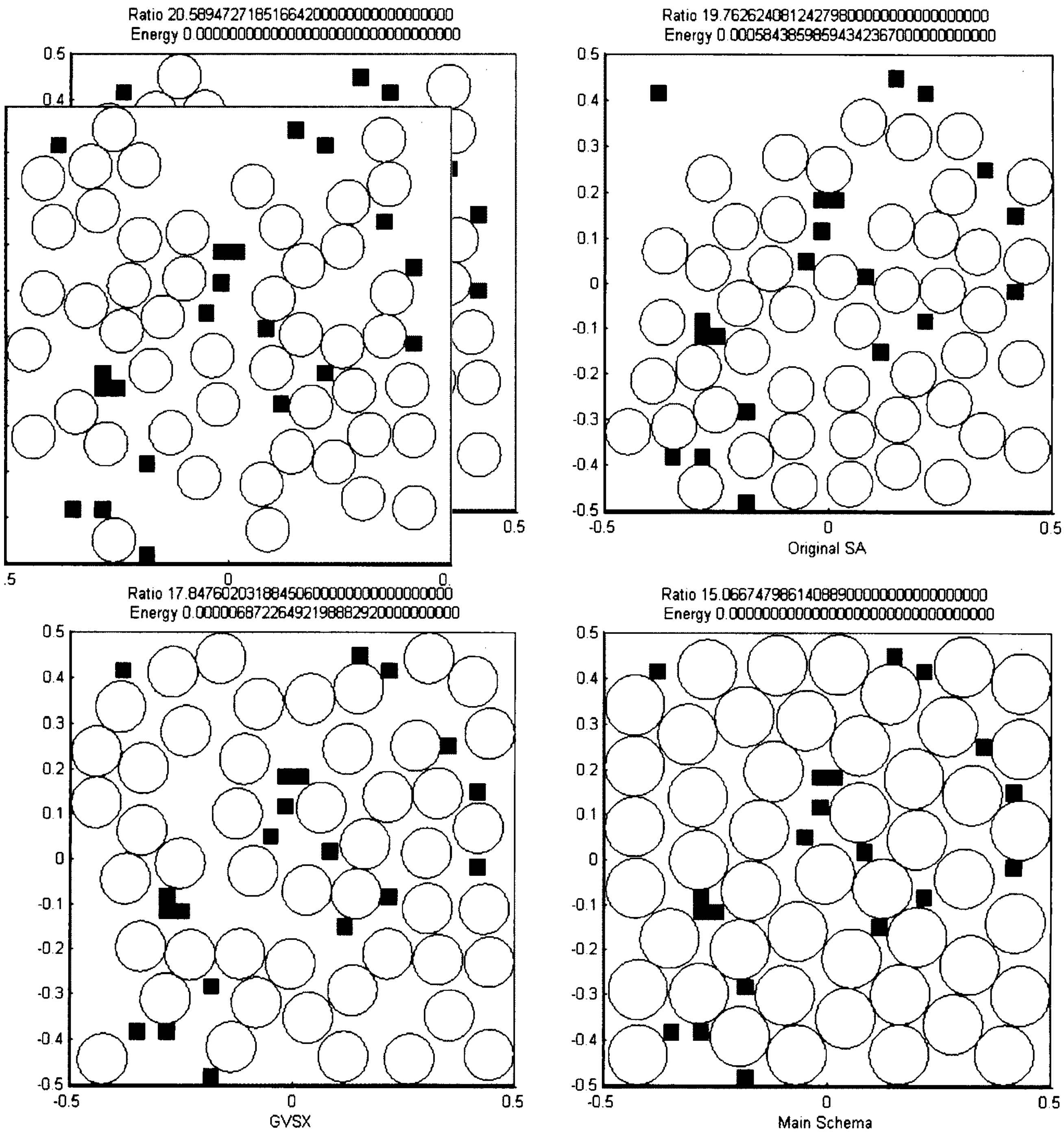


Figure 20: Experimental Result: 49 circle packing in a damaged square, [20/30^2].

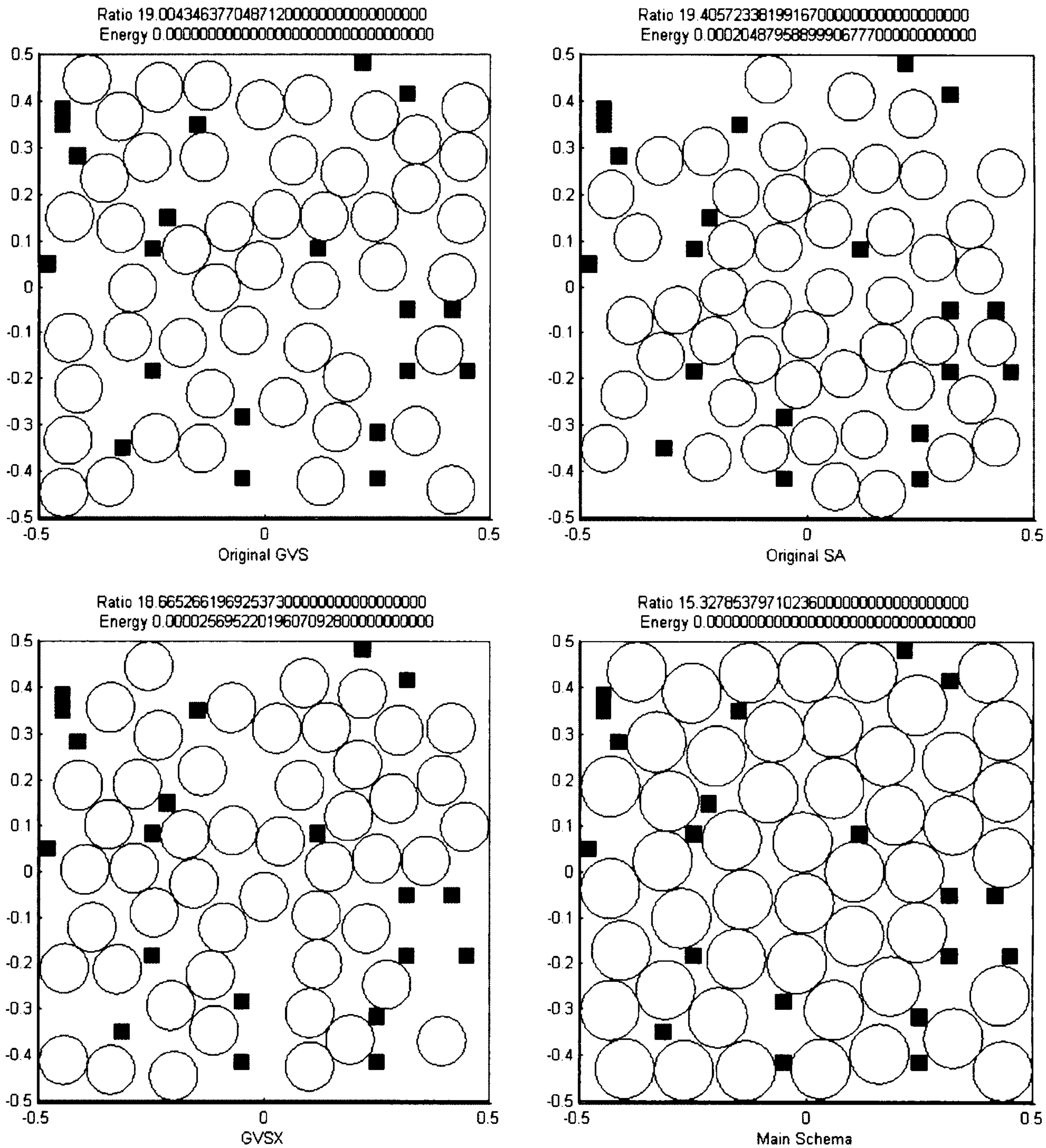


Figure 21: Experimental Result: 50 circle packing in a damaged square, [20/30^2].

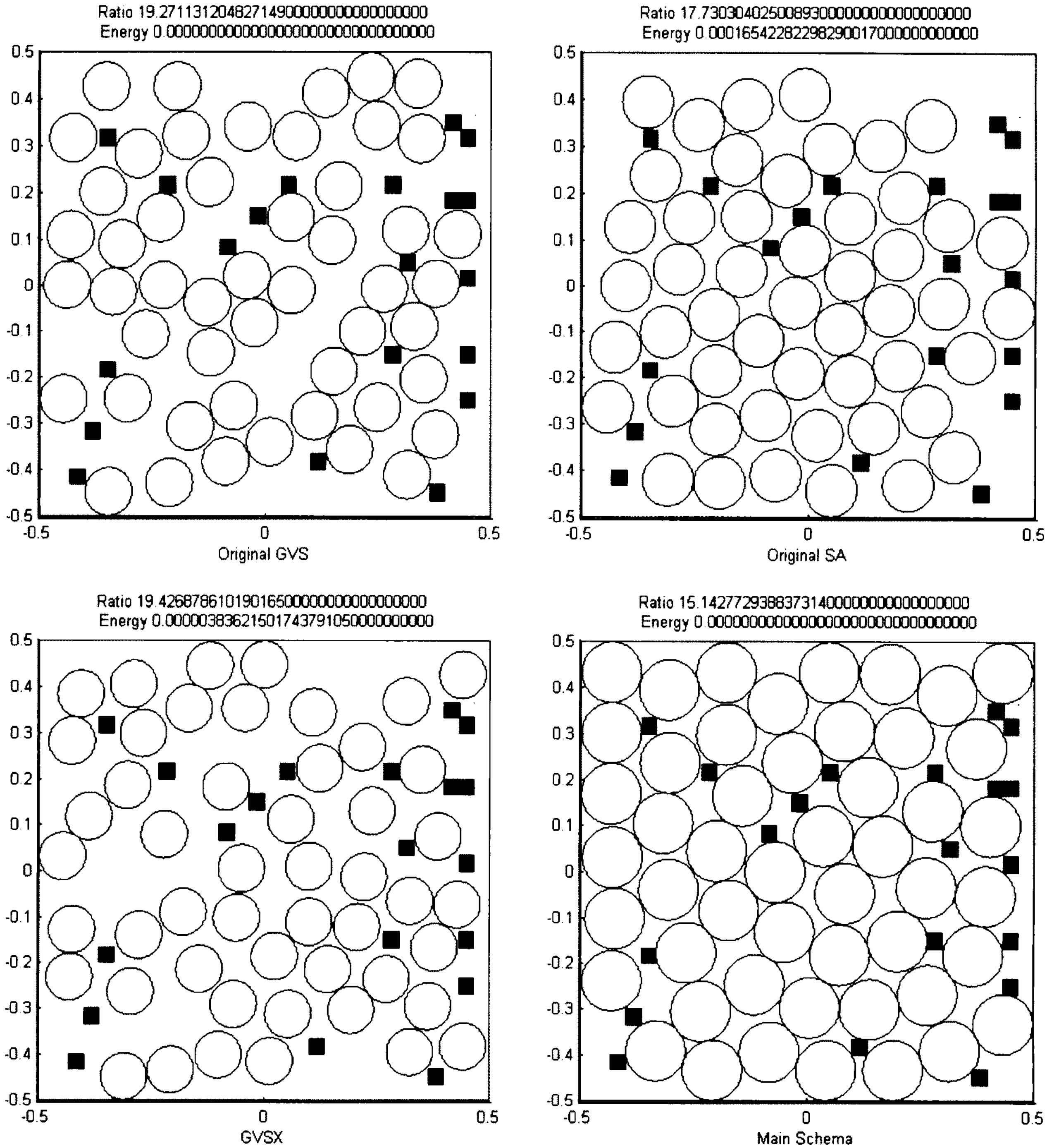


Figure 22: Experimental Result: 51 circle packing in a damaged square, $[20/30^2]$.

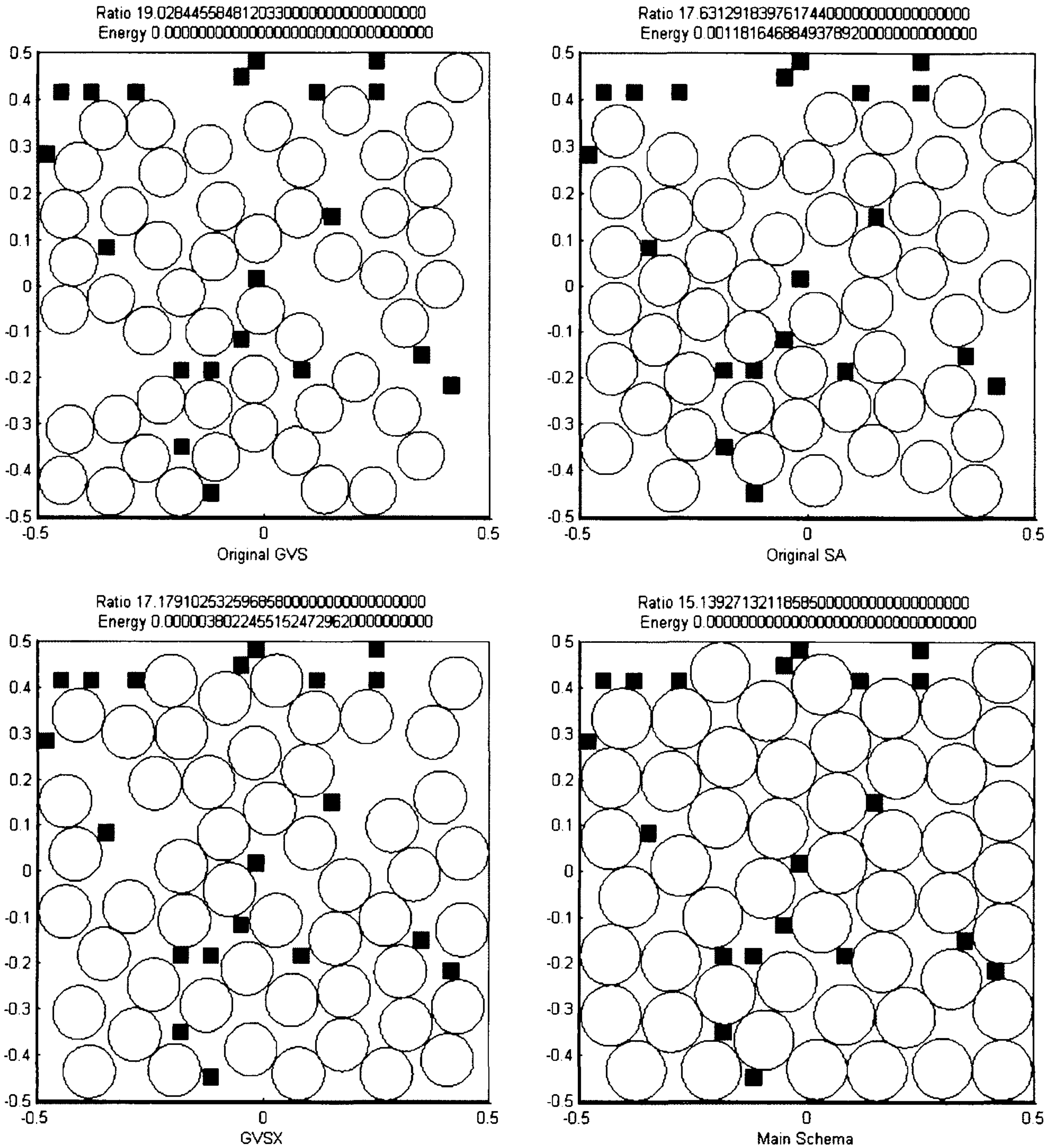


Figure 23: Experimental Result: 52 circle packing in a damaged square, [20/30^2].

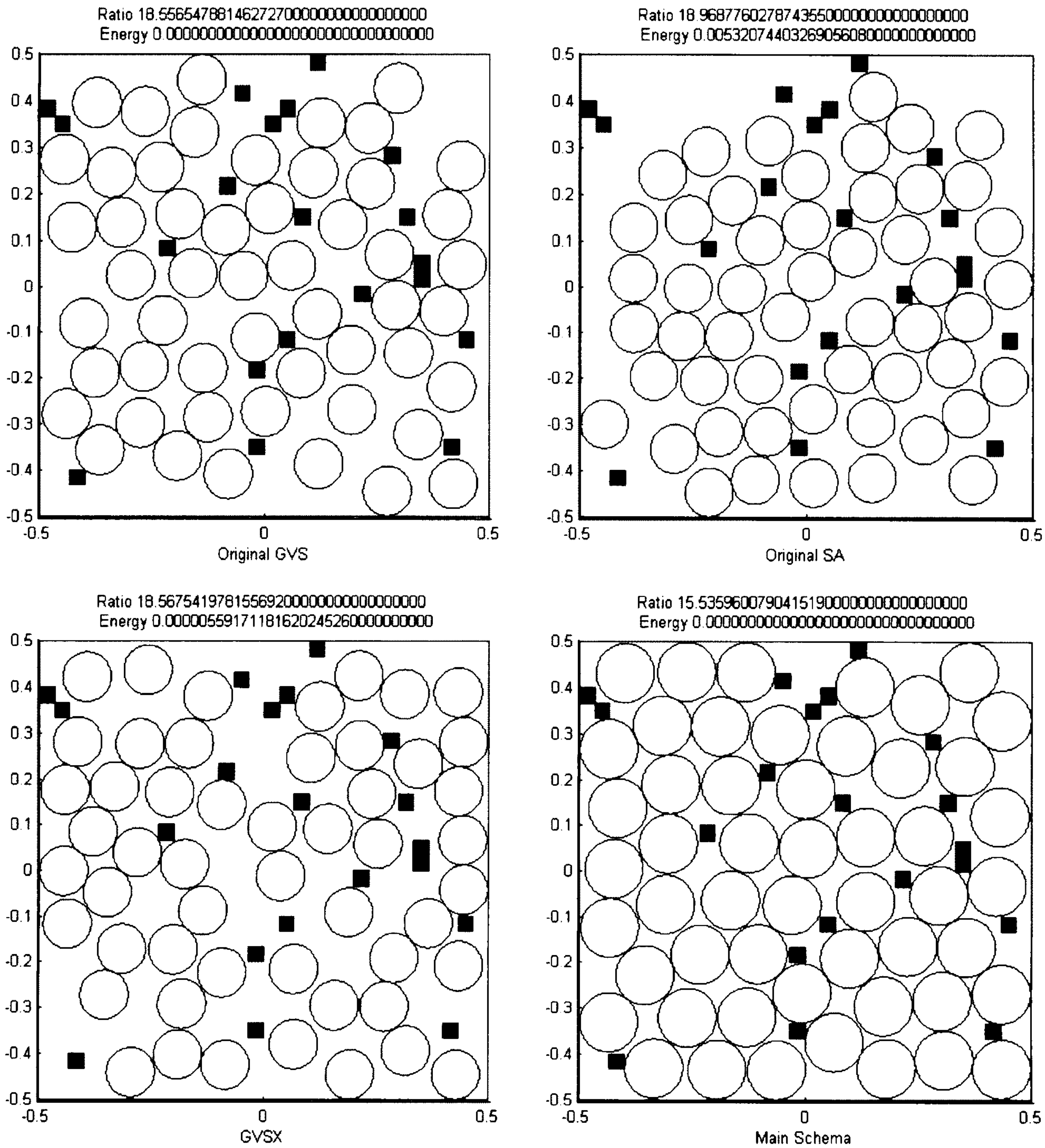


Figure 24: Experimental Result: 53 circle packing in a damaged square, [$20/30^2$].

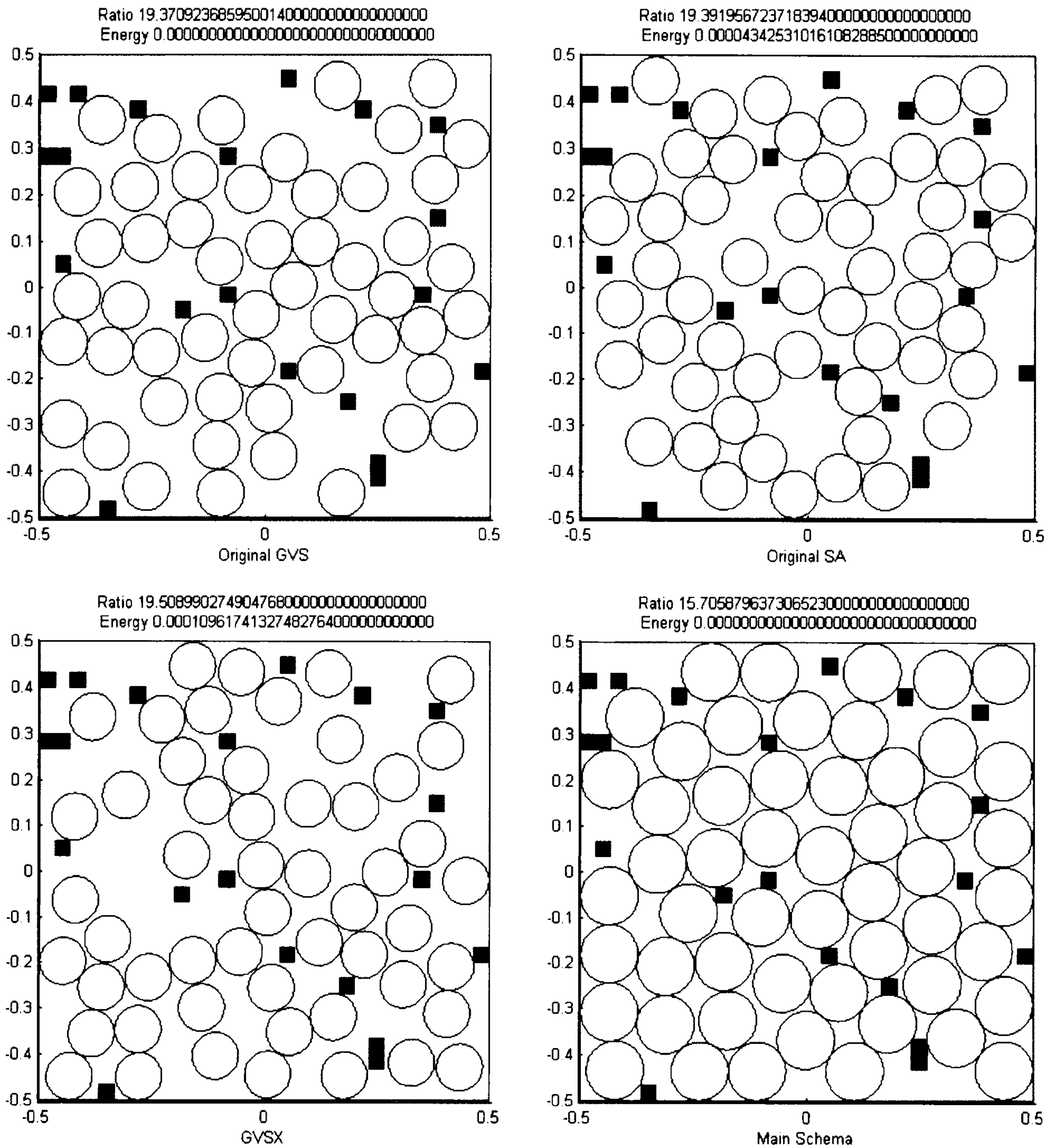


Figure 25: Experimental Result: 54 circle packing in a damaged square, $[20/30^2]$.

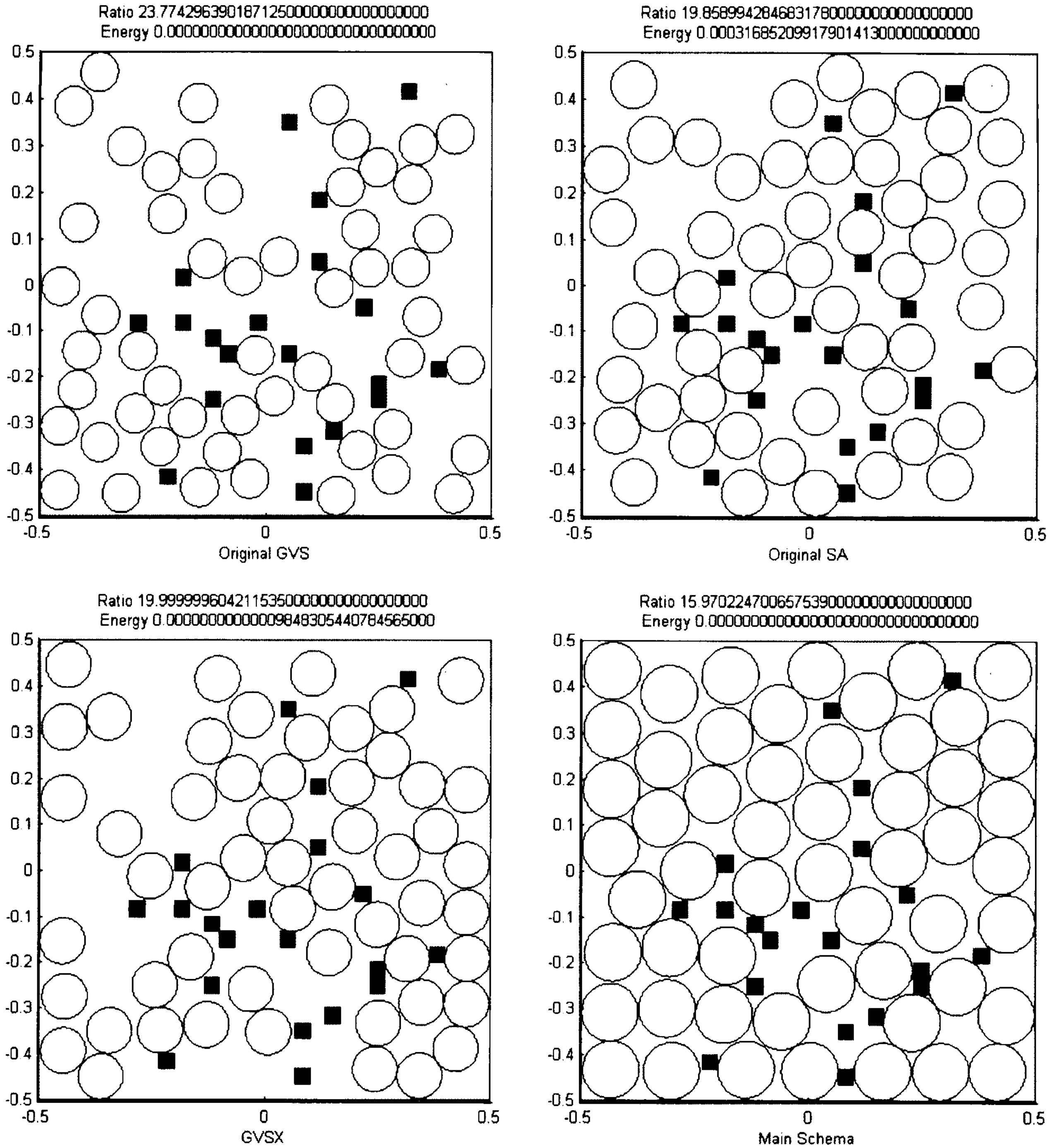


Figure 26: Experimental Result: 55 circle packing in a damaged square, [20/30^2].

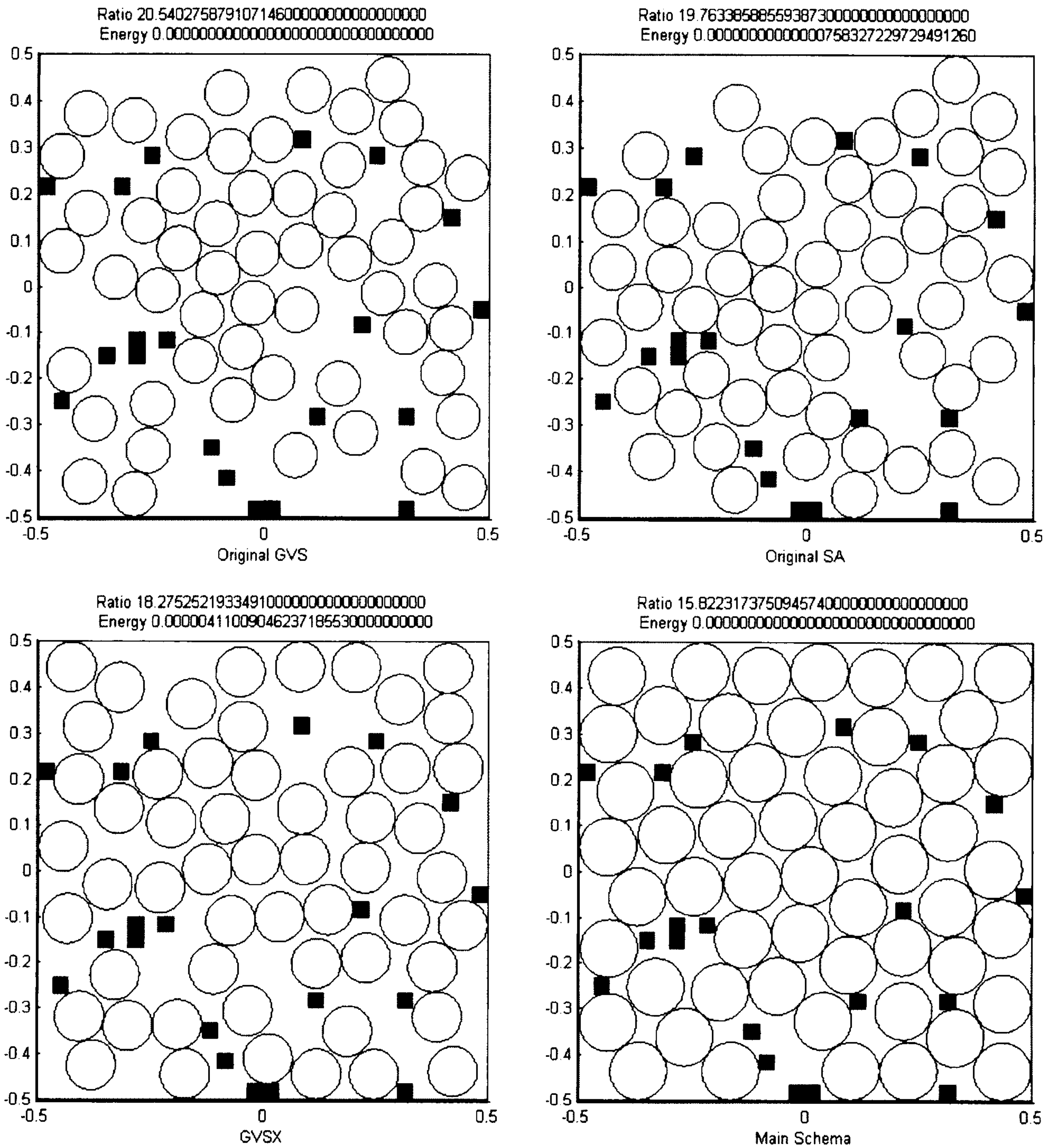


Figure 27: Experimental Result: 56 circle packing in a damaged square, $[20/30^2]$.

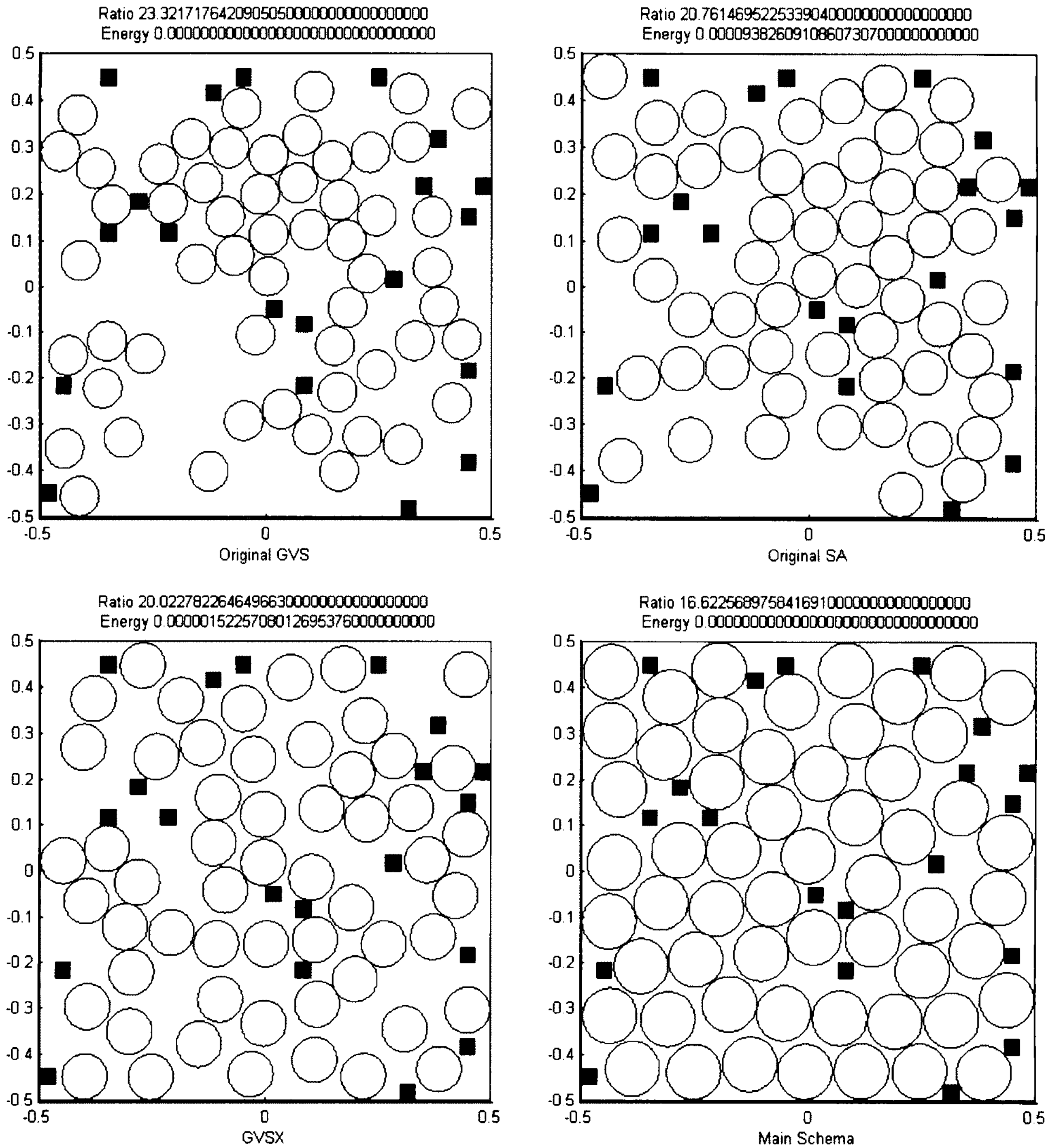


Figure 28: Experimental Result: 57 circle packing in a damaged square, $[20/30^2]$.

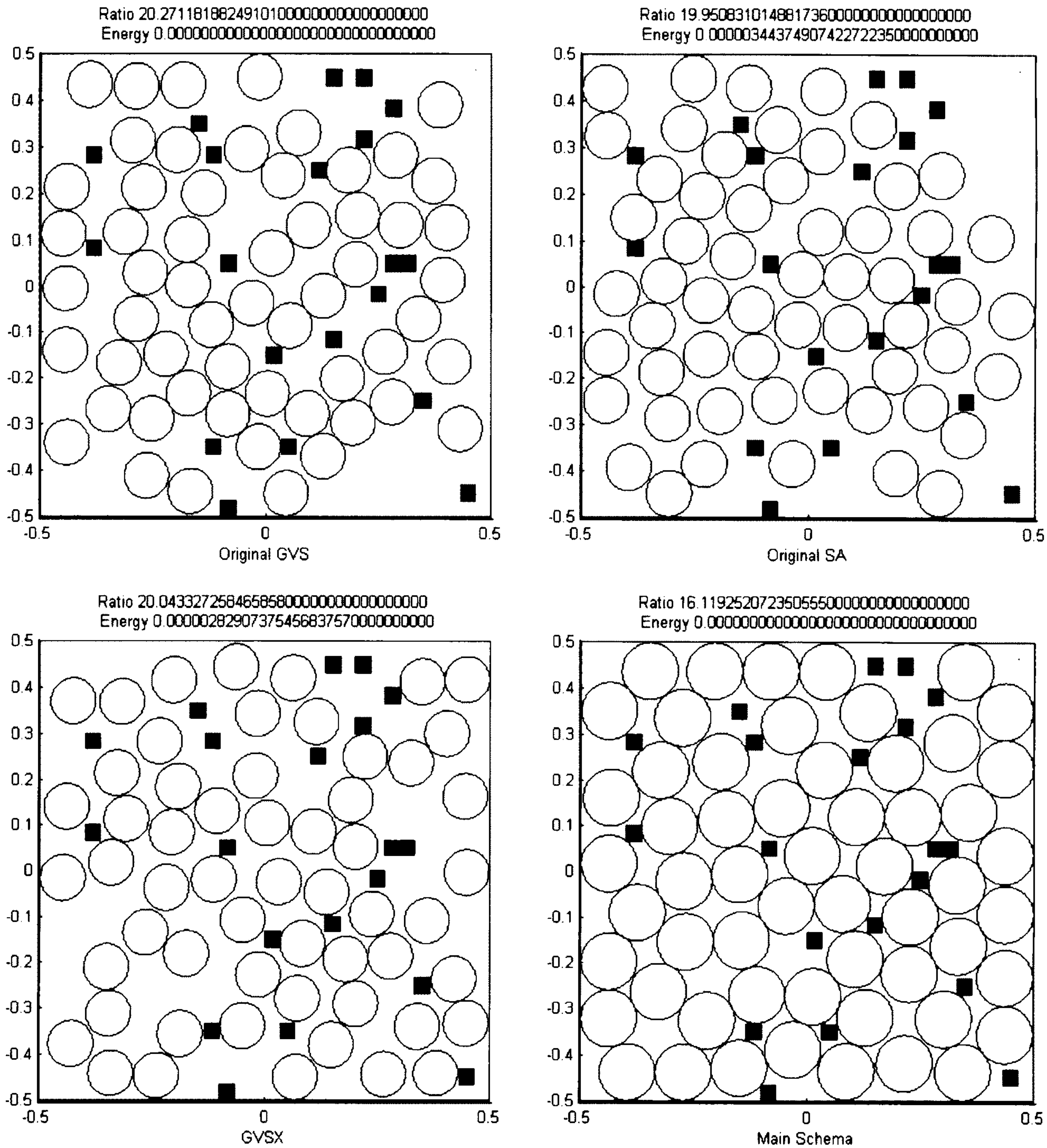


Figure 29: Experimental Result: 58 circle packing in a damaged square, $[20/30^2]$.

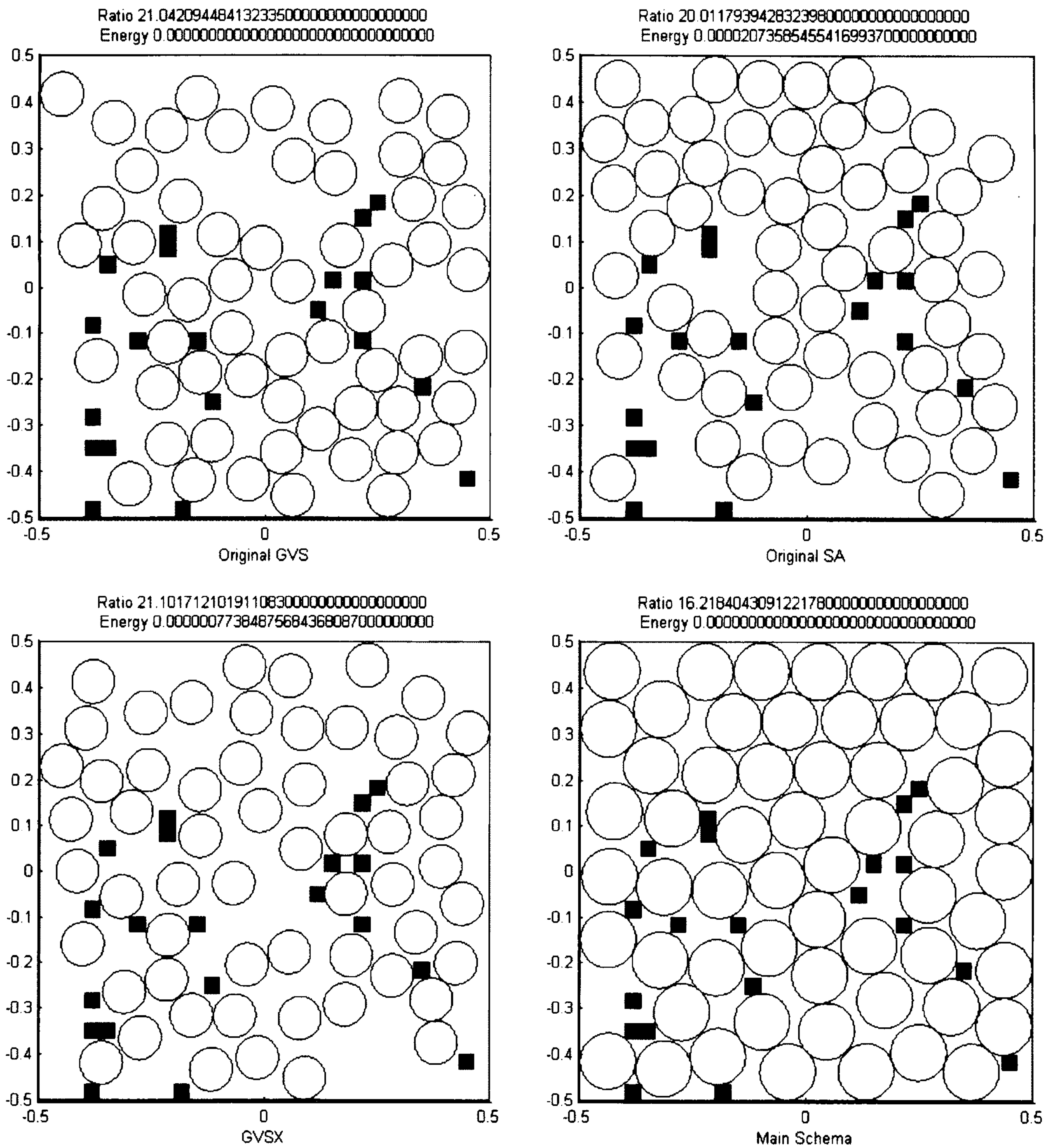


Figure 30: Experimental Result: 59 circle packing in a damaged square, [20/30^2].

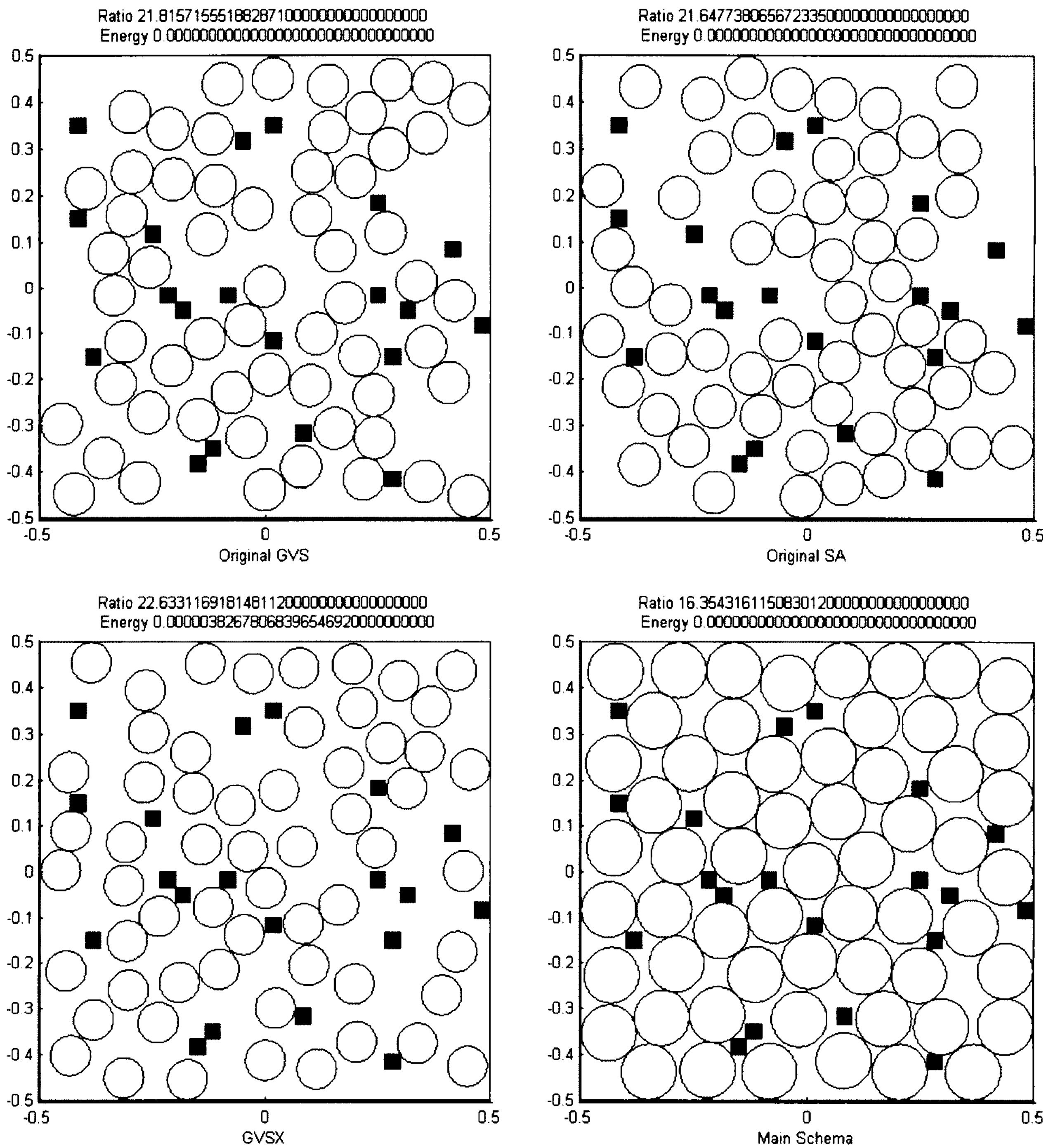


Figure 31: Experimental Result: 60 circle packing in a damaged square, $[20/30^2]$.

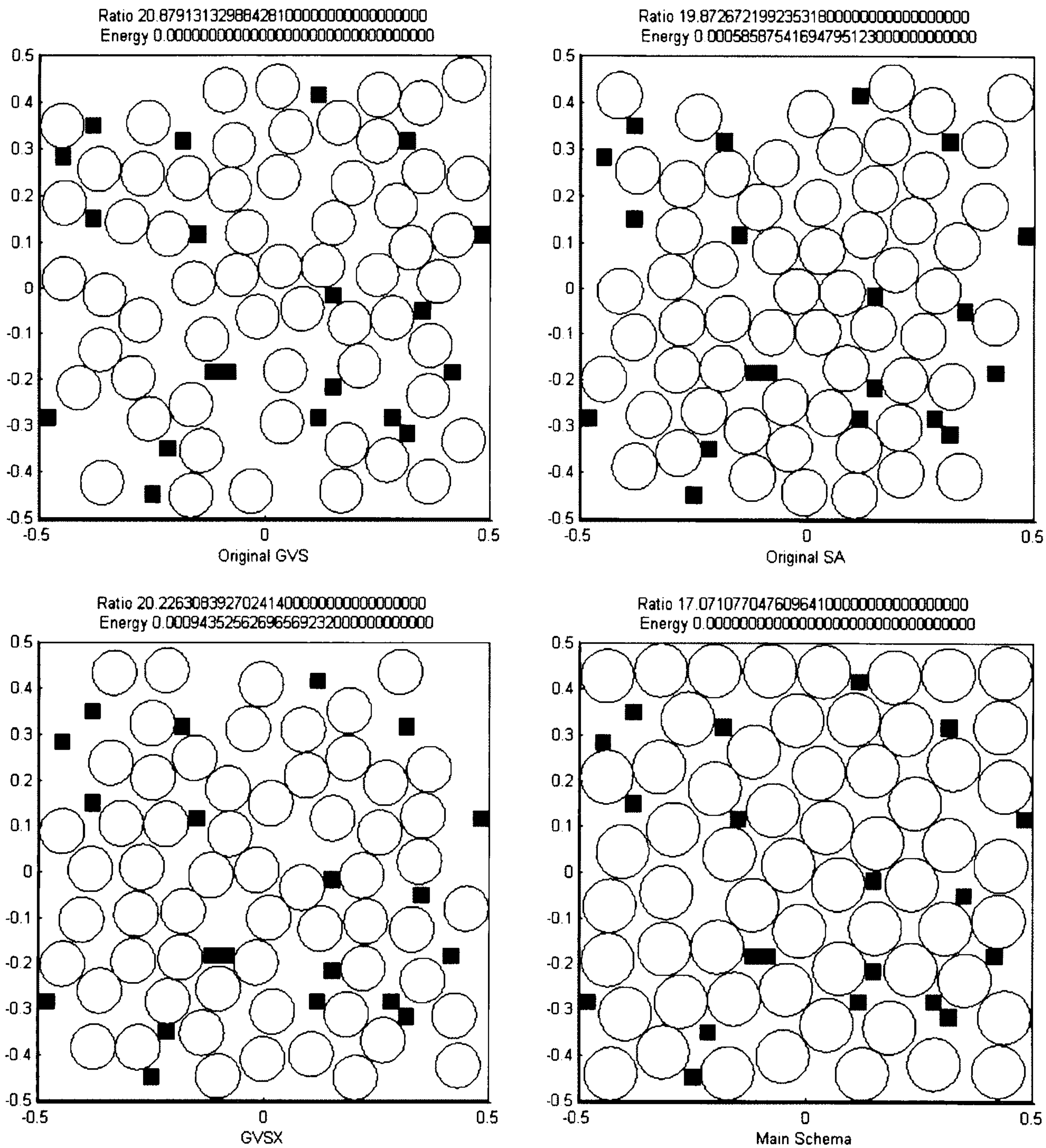


Figure 32: Experimental Result: 61 circle packing in a damaged square, [20/30^2].

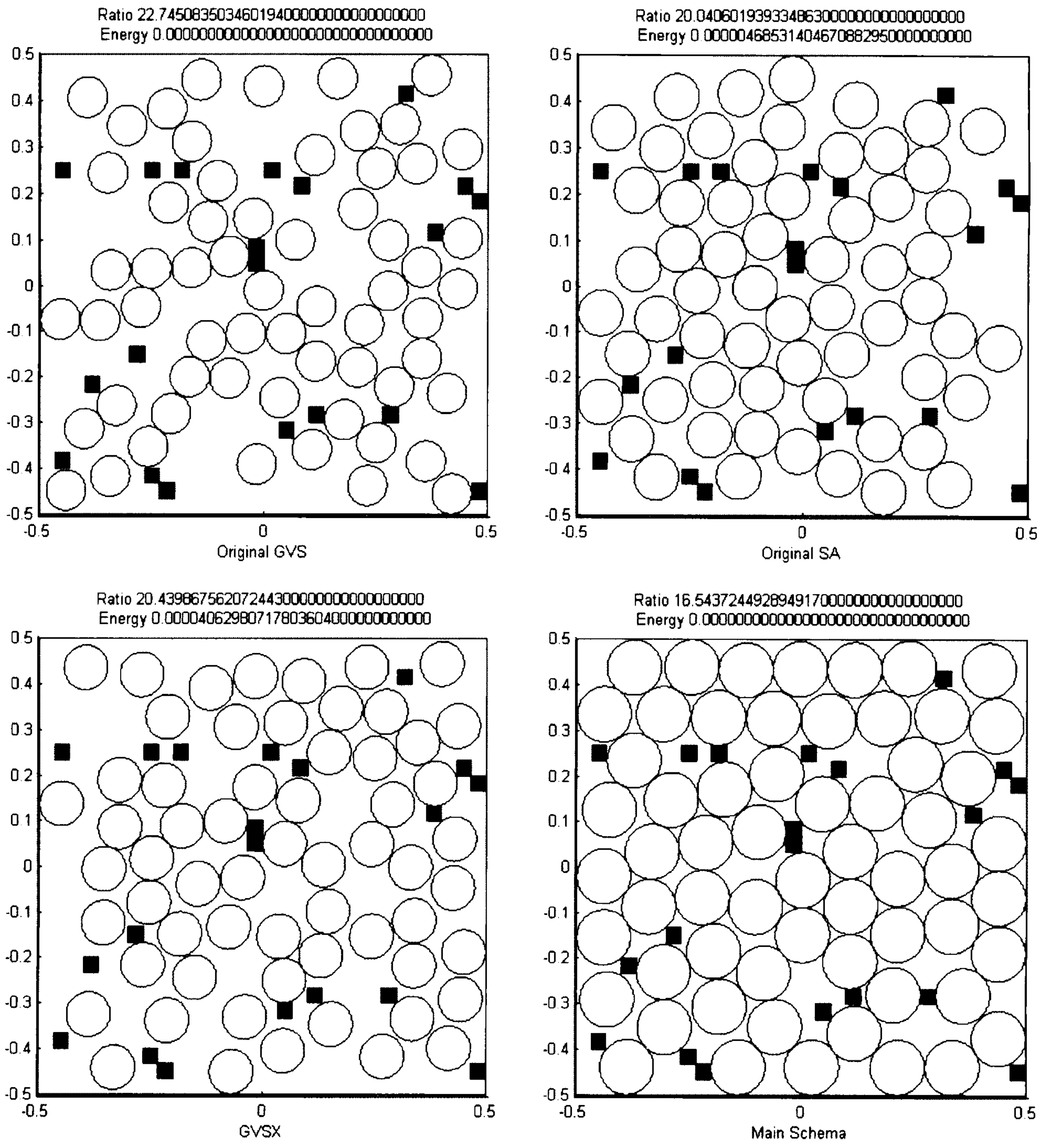


Figure 33: Experimental Result: 62 circle packing in a damaged square, $[20/30^2]$.

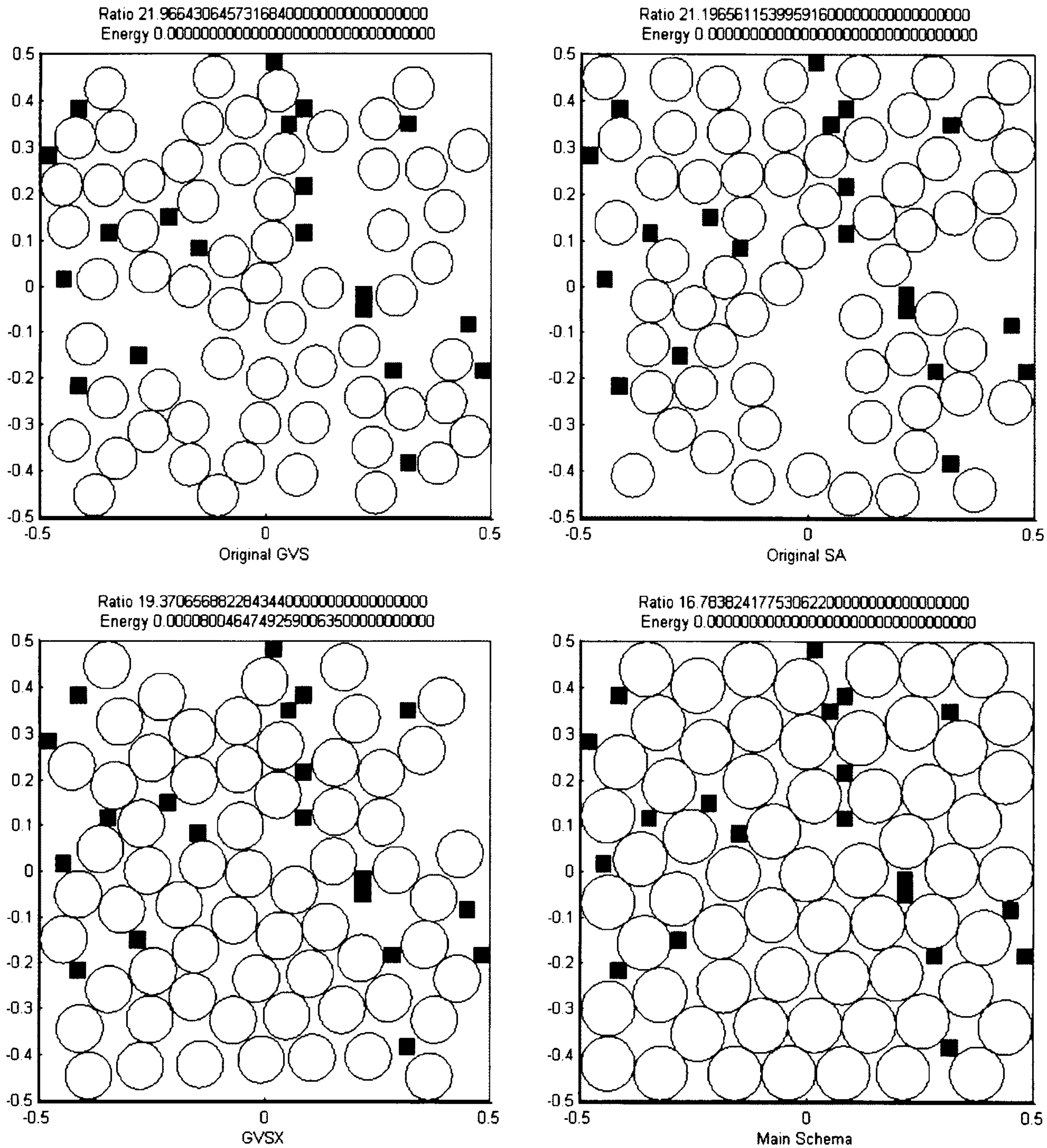


Figure 34: Experimental Result: 63 circle packing in a damaged square, $[20/30^2]$.

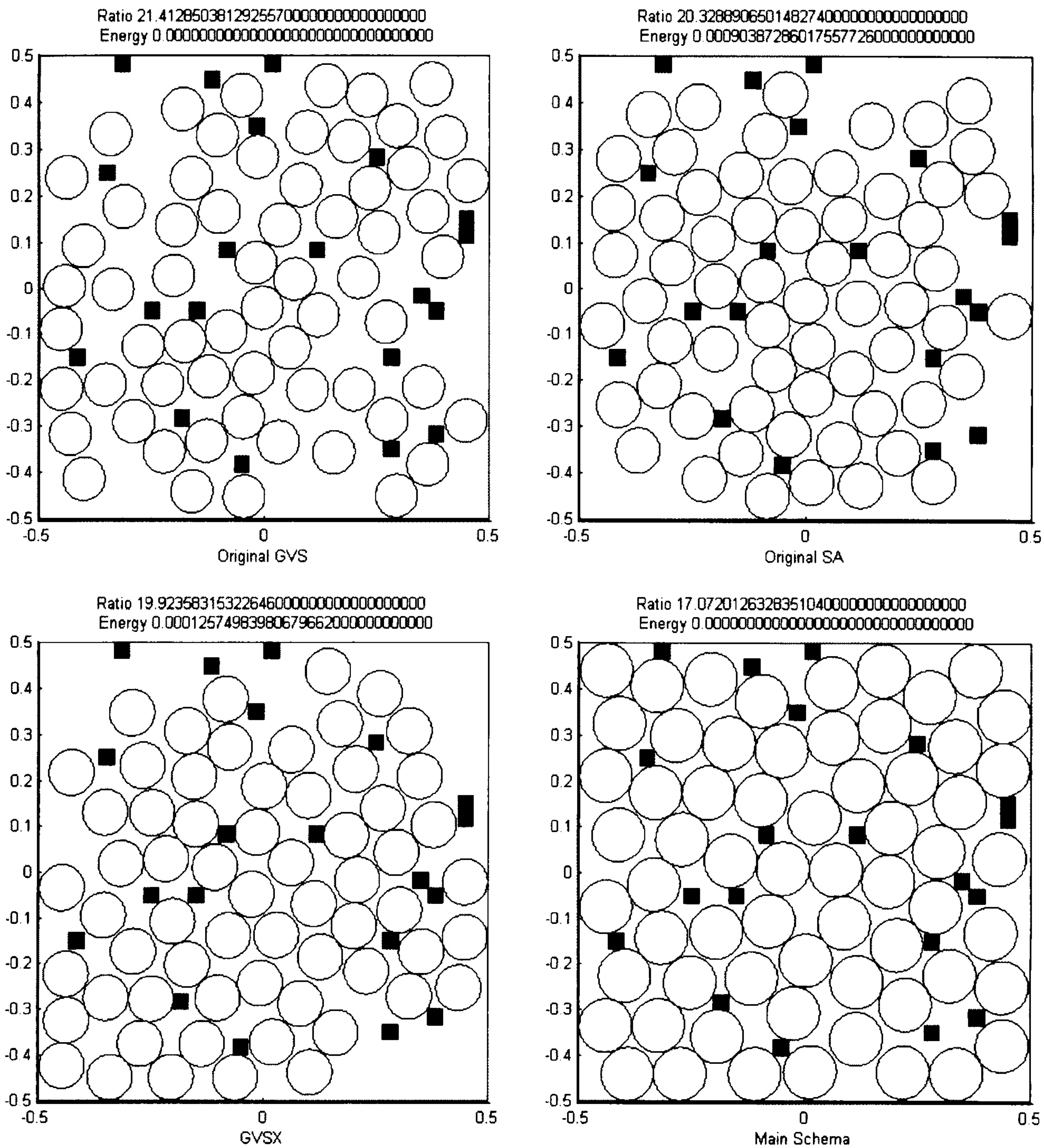


Figure 35: Experimental Result: 64 circle packing in a damaged square, $[20/30^2]$.

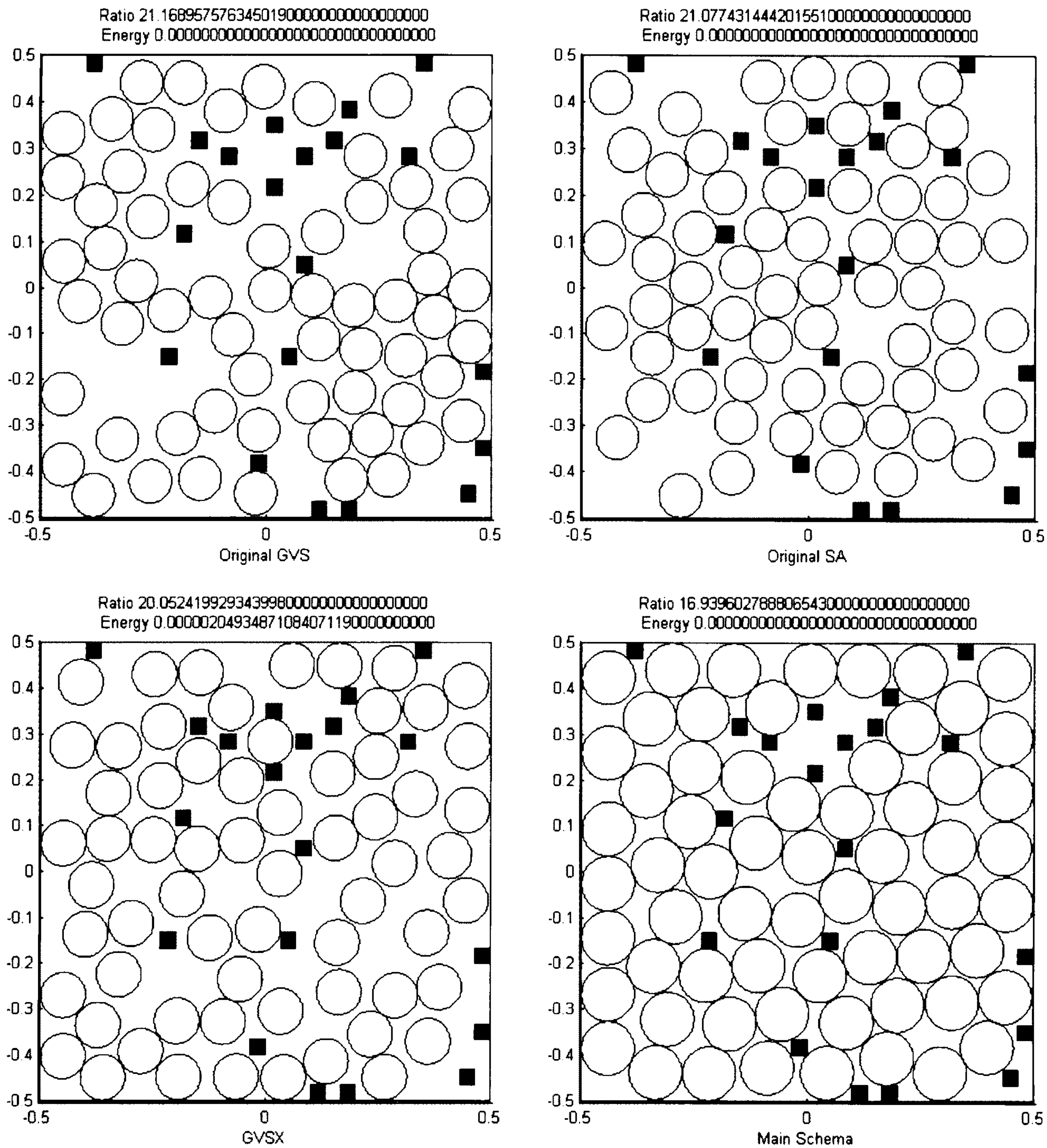


Figure 36: Experimental Result: 65 circle packing in a damaged square, [$20/30^2$].

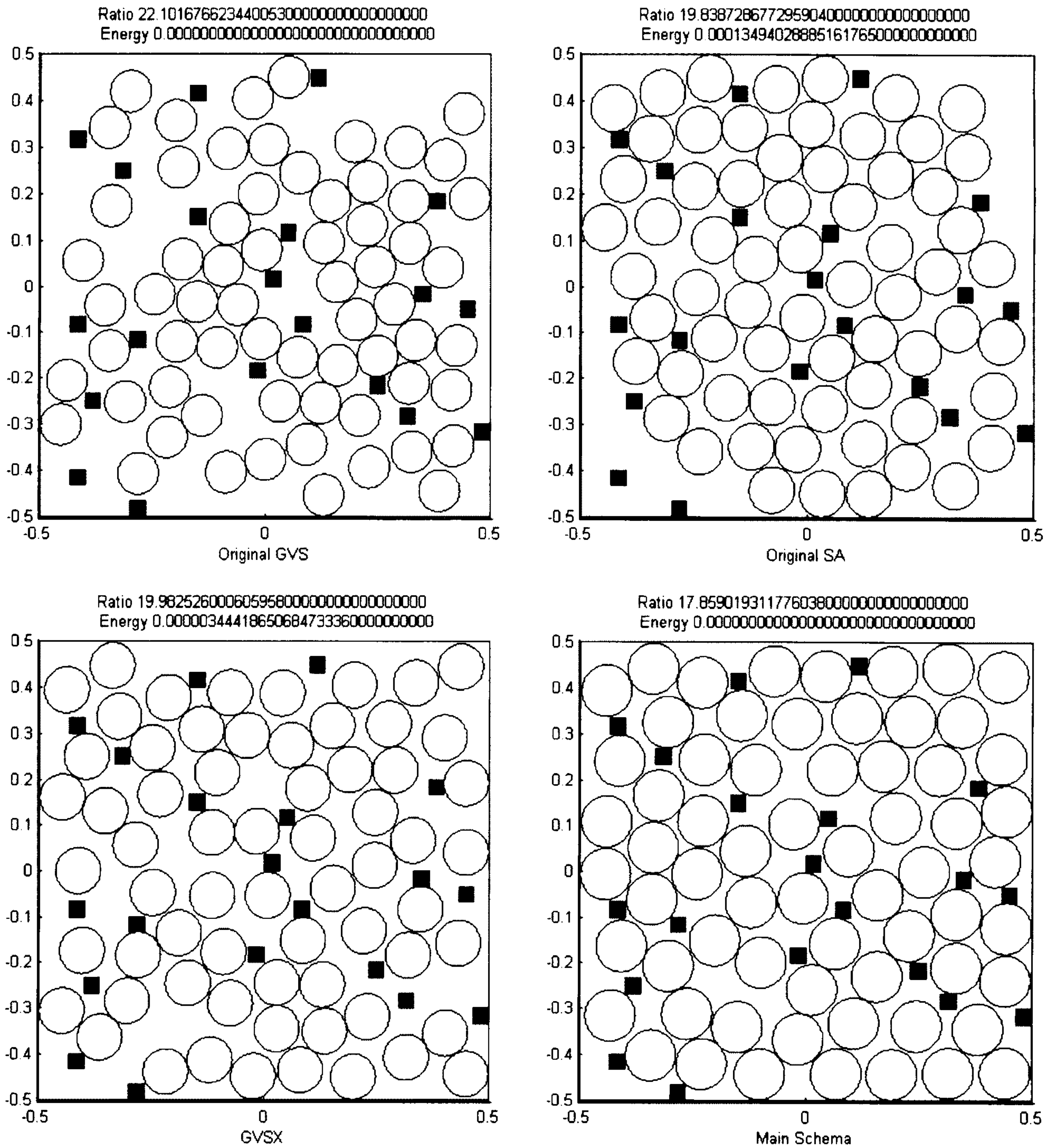


Figure 37: Experimental Result: 66 circle packing in a damaged square, $[20/30^2]$.

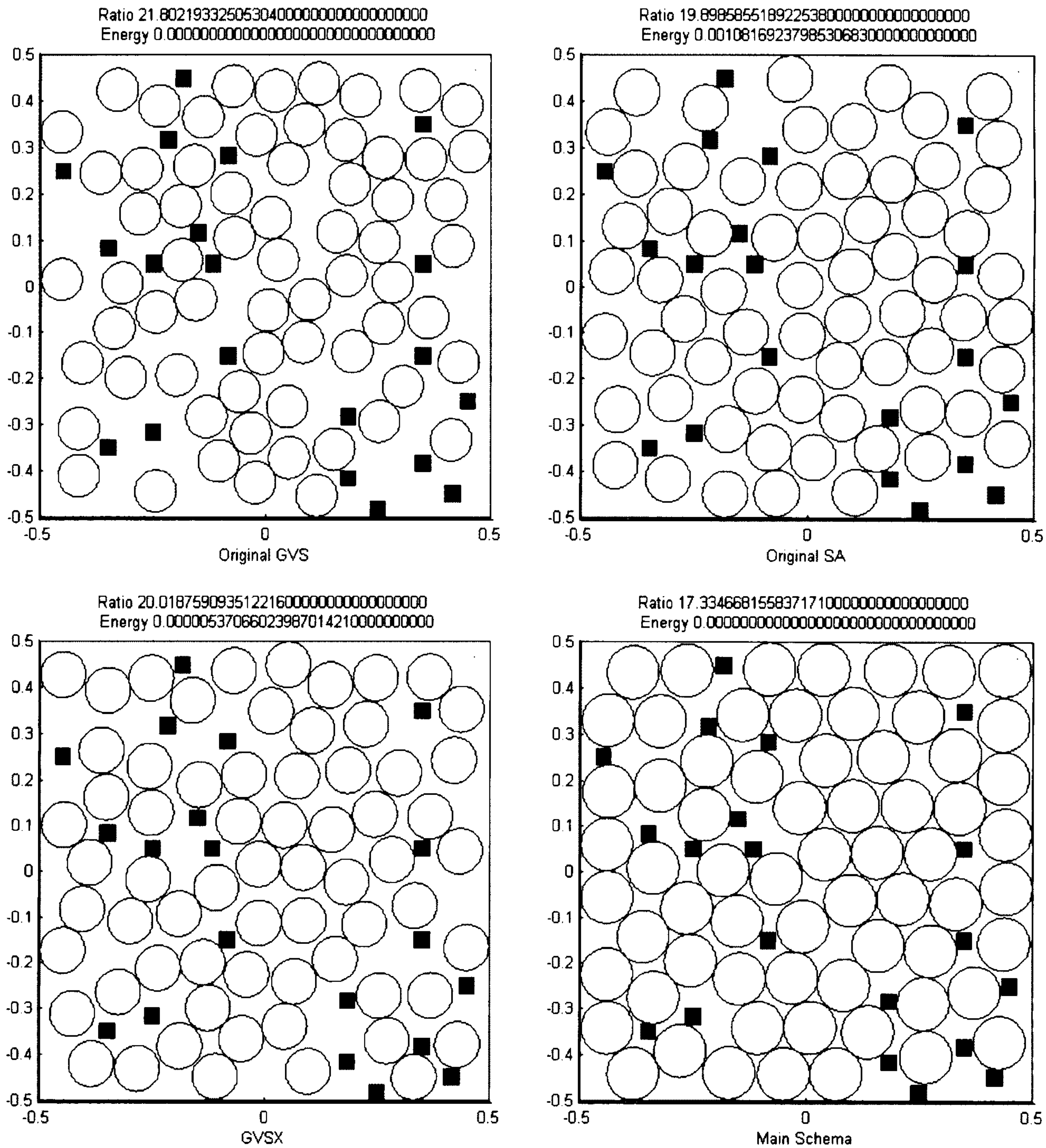


Figure 38: Experimental Result: 67 circle packing in a damaged square, $[20/30^2]$.

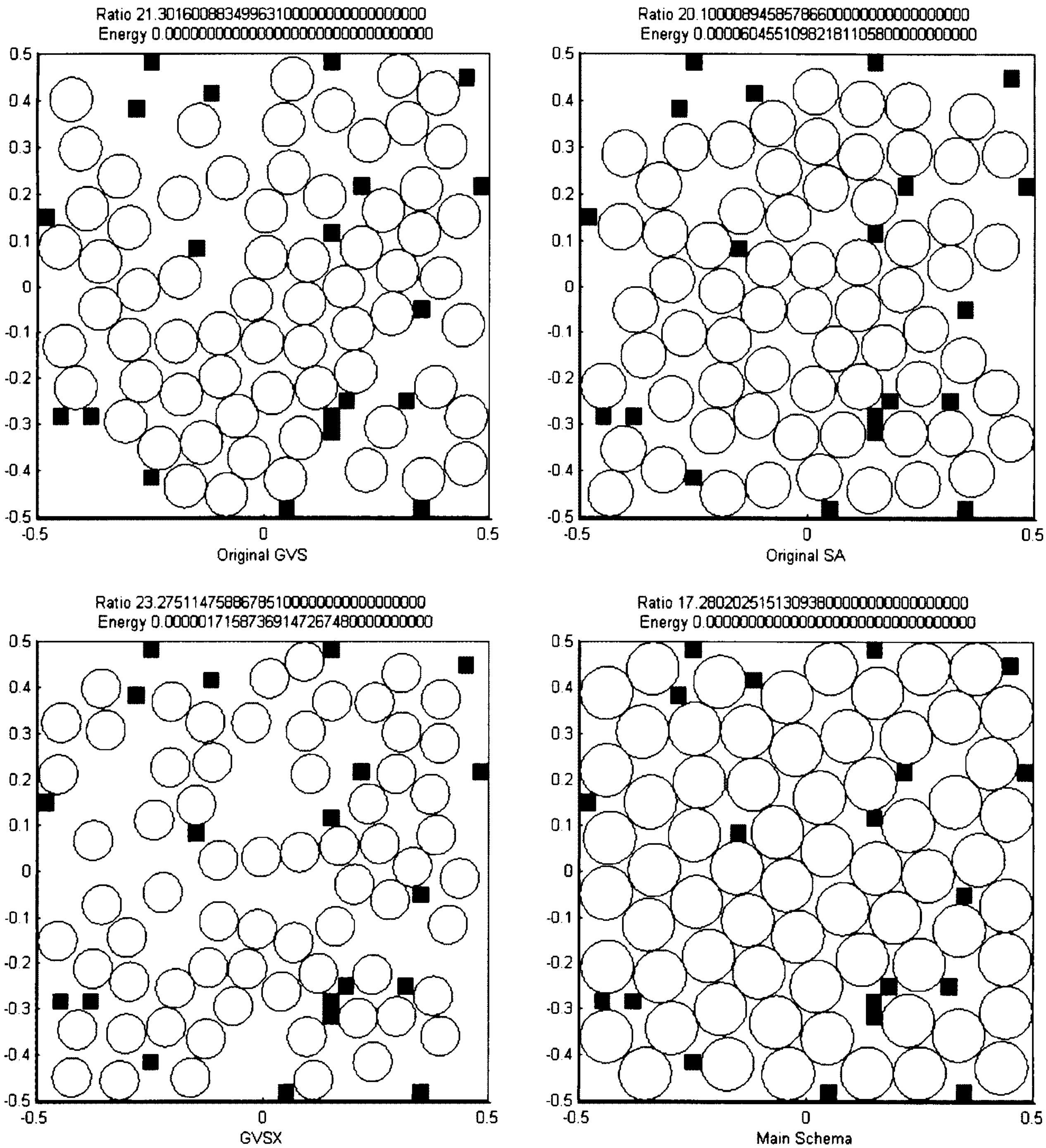
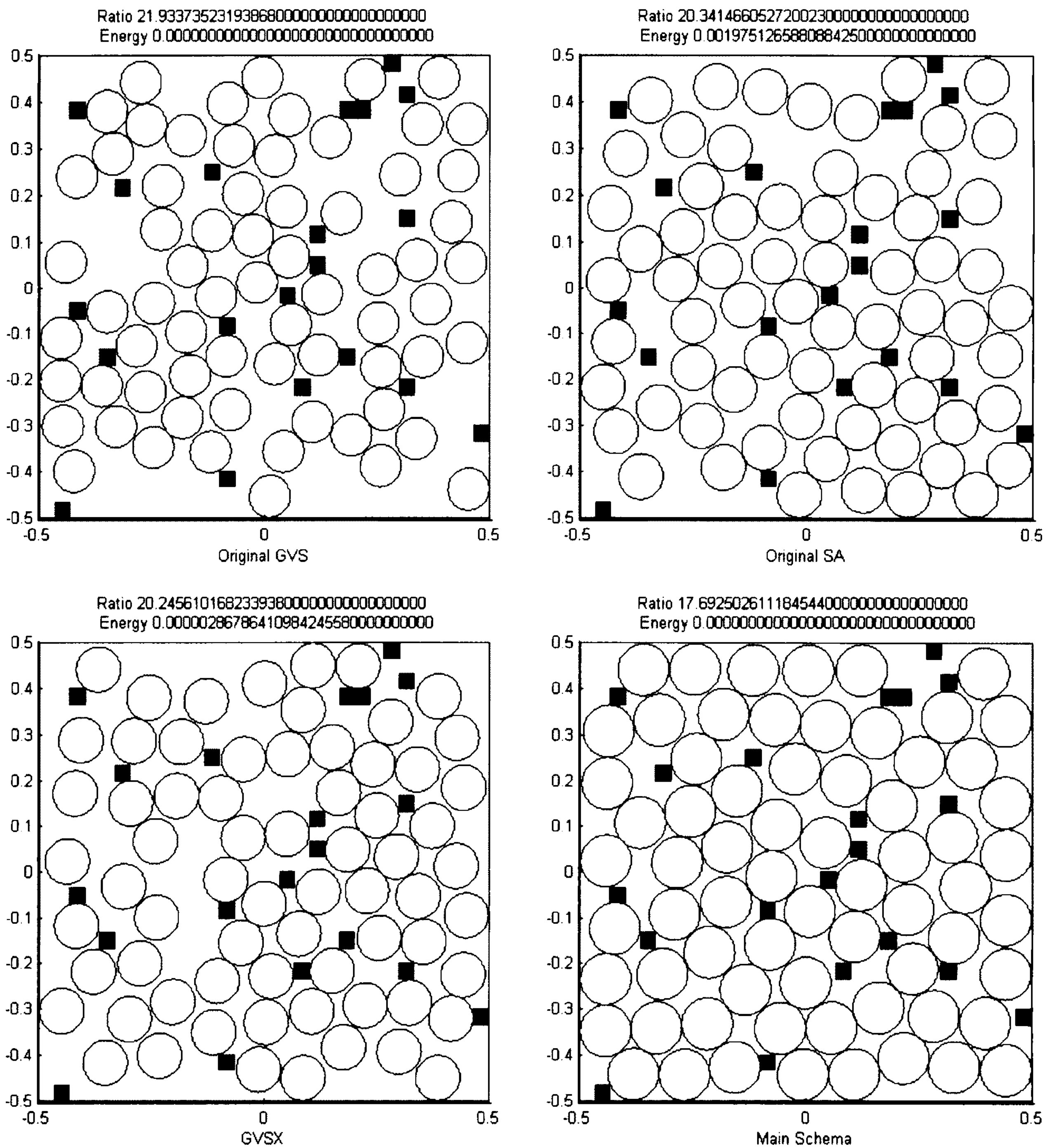


Figure 39: Experimental Result: 68 circle packing in a damaged square, [$20/30^2$].



Bibliography

[1] Bernardetta Addis. Packing circles in a square : new putative optima obtained via global optimization. 2005.

[2] J E Beasley and Mathematical Sciences. A heuristic for the circle packing problem with a variety of containers. pages 1 18, 2011.

[3] Ernesto G Birgin and Jan M Gentil. New and improved results for packing identical unitary radius circles within triangles , rectangles and strips . pages 1 20, 2009.

[4] Ernesto G. Birgin and Jan M. Gentil. New and improved results for packing identical unitary radius circles within triangles, rectangles and strips. *Computers & Operations Research*, 37(7):1318 1327, July 2010. ISSN 03050548. doi: 10.1016/j.cor.2009.09.017.

[5] LG Casado, I Garcia, PG Szabó, and T Csendes. Packing equal circles in a square ii.new results for up to 100 circles using the tamsass-pecs algorithm. In *Optimization Theory*, pages 207 224. Springer, 2001.

[6] Ignacio Castillo, Frank J. Kampas, and János D. Pintér. Solving circle packing problems by global optimization: Numerical results and industrial applications.

European Journal of Operational Research, 191(3):786 802, December 2008. ISSN 03772217. doi: 10.1016/j.ejor.2007.01.054.

[7] Ignacio Castillo, Frank J. Kampas, and János D. Pintér. Solving circle packing problems by global optimization: Numerical results and industrial applications. *European Journal of Operational Research*, 191(3):786 802, December 2008. ISSN 03772217. doi: 10.1016/j.ejor.2007.01.054.

[8] Hctor Corte. Matlab implementation of generic simulated annealling, 2010. URL `http://www.mathworks.com/matlabcentral/fileexchange/33109-simulated-annealing-optimization`.

[9] Alberto Costa. Valid constraints for the point packing in a square problem. *Discrete Applied Mathematics*, 161(18):2901 2909, December 2013. ISSN 0166218X. doi: 10.1016/j.dam.2013.06.008.

[10] Zhang De-fu. A personified annealing algorithm for circles packing problem 1. 31(4):4 9, 2005.

[11] Zhang De-fu. An improved heuristic recursive strategy based on genetic algorithm for the strip rectangular. 2007. doi: 10.1360/aas-007-0911.

[12] D.Kroon, 2009. URL `http://www.mathworks.com/matlabcentral/fileexchange/24301-finite-iterative-closest-point/content/fminlbfgs.m`.

[13] Shamil I. Galiev and Maria S. Lisafina. Linear models for the approximate solution of the problem of packing equal circles into a given domain. *European Journal of Operational Research*, 230(3):505 514, November 2013. ISSN 03772217. doi: 10.1016/j.ejor.2013.04.050.

[14] Marcus Gallagher. Investigating circles in a square packing problems as a realistic benchmark for continuous metaheuristic optimization algorithms. pages 1 10, 2009.

[15] Ronald L. Graham, Jeffrey C. Lagarias, Colin L. Mallows, Allan R. Wilks, and Catherine H. Yan. Apollonian circle packings: number theory. *Journal of Number Theory*, 100(1):1 45, May 2003. ISSN 0022314X. doi: 10.1016/S0022-314X(03)00015-5.

[16] Kun He, Danzeng Mo, Tao Ye, and Wenqi Huang. A coarse-to-fine quasi-physical optimization method for solving the circle packing problem with equilibrium constraints. *Computers & Industrial Engineering*, 66(4):1049 1060, December 2013. ISSN 03608352. doi: 10.1016/j.cie.2013.08.010.

[17] Kun He, Danzeng Mo, Tao Ye, and Wenqi Huang. A coarse-to-fine quasi-physical optimization method for solving the circle packing problem with equilibrium constraints. *Computers & Industrial Engineering*, 66(4):1049 1060, December 2013. ISSN 03608352. doi: 10.1016/j.cie.2013.08.010.

[18] M. Hifi and R. MHallah. A dynamic adaptive local search algorithm for the circular packing problem. *European Journal of Operational Research*, 183(3): 1280 1294, December 2007. ISSN 03772217. doi: 10.1016/j.ejor.2005.11.069.

[19] Mhand Hifi, Vangelis Th. Paschos, and Vassilis Zissimopoulos. A simulated annealing approach for the circular cutting problem. *European Journal of Operational Research*, 159(2):430 448, December 2004. ISSN 03772217. doi: 10.1016/S0377-2217(03)00417-X.

[20] Ignacio Hinostroza, Lorena Pradenas, and Víctor Parada. Board cutting from logs: Optimal and heuristic approaches for the problem of packing rectangles

in a circle. *International Journal of Production Economics*, 145(2):541 546, October 2013. ISSN 09255273. doi: 10.1016/j.ijpe.2013.04.047.

[21] Wenqi Huang and Tao Ye. Greedy vacancy search algorithm for packing equal circles in a square. *Operations Research Letters*, 38(5):378 382, September 2010. ISSN 01676377. doi: 10.1016/j.orl.2010.07.004.

[22] Wenqi Huang and Tao Ye. Global optimization method for finding dense packings of equal circles in a circle. *European Journal of Operational Research*, 210 (3):474 481, May 2011. ISSN 03772217. doi: 10.1016/j.ejor.2010.11.020.

[23] Stefan Jakobs. On genetic algorithms for the packing of polygons. *European Journal of Operational Research*, 88(1):165 181, January 1996. ISSN 03772217. doi: 10.1016/0377-2217(94)00166-9.

[24] Gelatt C.D. Vecchi M.P. Kirkpatrick, S. Optimization by simulated annealing.science. pages 220, 671 680, 1983.

[25] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *SCIENCE*, 220(4598):671 680, 1983.

[26] Jingfa Liu, Gang Li, Duanbing Chen, Wenjie Liu, and Yali Wang. Two-dimensional equilibrium constraint layout using simulated annealing. *Computers & Industrial Engineering*, 59(4):530 536, November 2010. ISSN 03608352. doi: 10.1016/j.cie.2010.06.009.

[27] Marco Locatelli and Ulrich Raber. Packing equal circles in a square: a deterministic global optimization approach. *Discrete Applied Mathematics*, 122(1-3): 139 166, October 2002. ISSN 0166218X. doi: 10.1016/S0166-218X(01)00359-6.

[28] Andrea Lodi, Silvano Martello, and Michele Monaci. Two-dimensional packing problems: A survey. *European Journal of Operational Research*, 141(2):241 252, September 2002. ISSN 03772217. doi: 10.1016/S0377-2217(02)00123-6.

[29] C.O. López and J.E. Beasley. Packing unequal circles using formulation space search. *Computers & Operations Research*, 40(5):1276 1288, May 2013. ISSN 03050548. doi: 10.1016/j.cor.2012.11.022.

[30] Mihály Csaba Markót. Interval methods for verifying structural optimality of circle packing configurations in the unit square. *Journal of Computational and Applied Mathematics*, 199(2):353 357, February 2007. ISSN 03770427. doi: 10.1016/j.cam.2005.08.039.

[31] MathWorks. fminunc, 2014. URL `http://www.mathworks.com/help/optim/ug/fminunc.html`.

[32] Nenad Mladenović, Frank Plastria, and Dragan Urošević. Reformulation descent applied to circle packing problems. *Computers & Operations Research*, 32(9):2419 2434, September 2005. ISSN 03050548. doi: 10.1016/j.cor.2004.03.010.

[33] Michael Mollard and Charles Payan. Some progress in the packing of equal circles in a square. *Discrete Mathematics*, 84(3):303 307, October 1990. ISSN 0012365X. doi: 10.1016/0012-365X(90)90135-5.

[34] Shuhei Morinaga, Hidenori Ohta, and Mario Nakamori. An algorithm for the circle-packing problem via extended sequence-pair with nonlinear optimization. II:23 25, 2013.

[35] Jorge Nocedal and Stephen J Wright. *Numerical Optimization 2nd Edition*. Springer.

[36] Kari J Nurmela. Optimal packings of equal circles in a square 1 introduction 2 a computer-aided method for optimality proofs.

[37] R Peikert, D Wurtz, M Monagan, C De Groot, R Milano, and G Valette. Packing circles in a square : A review and new results. 1989.

[38] I. Peterson. Cracking kepler's sphere-packing problem. 154:103, 1998.

[39] Peter N. Saeta. The metropolis algorithm. 2011.

[40] Boaz Barak Sanjeev Arora. *Computational Complexity: A Modern Approach*. Cambridge University Press Cambridge, 2009. ISBN 9780521424264.

[41] E. Specht. High density packings of equal circles in rectangles with variable aspect ratio. *Computers & Operations Research*, 40(1):58 69, January 2013. ISSN 03050548. doi: 10.1016/j.cor.2012.05.011.

[42] E. Specht. High density packings of equal circles in rectangles with variable aspect ratio. *Computers & Operations Research*, 40(1):58 69, January 2013. ISSN 03050548. doi: 10.1016/j.cor.2012.05.011.

[43] Eckard Specht. Packing up to 200 equal circles in a square. pages 1 17, 1960.

[44] Eckard Specht. The best known packings of equal circles in a square. 2013. URL `http://www.packomania.com`.

[45] PG Szabó and Eckard Specht. Packing up to 200 equal circles in a square. *Models and Algorithms for Global Optimization*, 2007.

[46] Vassilios E. Theodoracatos and James L. Grimsley. The optimal packing of arbitrarily-shaped polygons using simulated annealing and polynomial-time cooling schedules. *Computer Methods in Applied Mechanics and Engineering*,

125(1-4):53 70, September 1995. ISSN 00457825. doi: 10.1016/0045-7825(95)00795-3.

[47] Huaiqing Wang, Wenqi Huang, Quan Zhang, and Dongming Xu. An improved algorithm for the packing of unequal circles within a larger containing circle. *European Journal of Operational Research*, 141(2):440 453, September 2002. ISSN 03772217. doi: 10.1016/S0377-2217(01)00241-7.

[48] Huaiqing Wang, Wenqi Huang, Quan Zhang, and Dongming Xu. An improved algorithm for the packing of unequal circles within a larger containing circle. *European Journal of Operational Research*, 141(2):440 453, September 2002. ISSN 03772217. doi: 10.1016/S0377-2217(01)00241-7.

[49] Tae-Sang Chung John Morris John Whiley Won Y. Yang, Wenwu Cao and Sons. Applied numerical methods using matlab. 2005.

[50] W.Y. Yang, W. Cao, T.S. Chung, and J. Morris. *Applied Numerical Methods Using MATLAB*. Wiley, 2005. ISBN 9780471705185.

[51] De-fu Zhang and An-sheng Deng. An effective hybrid algorithm for the problem of packing circles into a larger containing circle. *Computers & Operations Research*, 32(8):1941 1951, August 2005. ISSN 03050548. doi: 10.1016/j.cor.2003.12.006.

[52] Chuanming Zong. Packing, covering and tiling in two-dimensional spaces. *Expositiones Mathematicae*, December 2013. ISSN 07230869. doi: 10.1016/j.exmath.2013.12.002.