3D Packing of Balls in Different Containers by VNS

A thesis submitted for the degree of

Doctor of Philosophy

by

Abdulaziz M. Alkandari

Supervisor

Nenad Mladenovic

Department Mathematical Sciences
School of Information Systems, Computing and Mathematics
Brunel University, London

June 2013

Abstract

In real world applications such as the transporting of goods products, packing is a major issue. Goods products need to be packed such that the smallest space is wasted to achieve the maximum transportation efficiency. Packing becomes more challenging and complex when the product is circular/spherical. This thesis focuses on the best way to pack three-dimensional unit spheres into the smallest spherical and cubical space. Unit spheres are considered in lieu of non-identical spheres because the search mechanisms are more difficult in the latter set up and any improvements will be due to the search mechanism not to the ordering of the spheres. The two-unit sphere packing problems are solved by approximately using a variable neighborhood search (VNS) hybrid heuristic. A general search framework belonging to the Artificial Intelligence domain, the VNS offers a diversification of the search space by changing neighborhood structures and intensification by thoroughly investigating each neighborhood. It is flexible, easy to implement, adaptable to both continuous and discrete optimization problems and has been use to solve a variety of problems including large-sized real-life problems. Its runtime is usually lower than other meta heuristic techniques. A tutorial on the VNS and its variants along with recent applications and areas of applicability of each variant. Subsequently, this thesis considers several variations of VNS heuristics for the two problems at hand, discusses their individual efficiencies and effectiveness, their convergence rates and studies their robustness. It highlights the importance of the hybridization which yields near global optima with high precision and accuracy, improving many best-known solutions indicate matching some, and improving the precision and accuracy of others.

Keywords: variable neighborhood search, sphere packing, three-dimensional packing, meta heuristic, hybrid heuristics, multiple start heuristics.

Declaration of Originality

I hereby certify that the work presented in this thesis is my original research and has not been presented for a higher degree at any other university or institute.

Signed:________________________ Dated: _________________

Abdulaziz Alkandari

Acknowledgements

First I would like to thank my God, whose response has always helped me and given me the strength to complete my work. I express my heartfelt gratitude to all those who made it possible for me to complete this thesis. I thank the Department of Mathematics for providing me with the most recent programs in my research, for giving me the requisite permissions to access the relevant articles in my work and to use the departmental facilities.

I am deeply indebted to my supervisor, Dr. Nenad Mladenovic, whose help, stimulating suggestions and encouragement helped me throughout my research as well as the writing of this thesis.

I thank Dr. Rym M'Halah, for patiently explaining most of the software tools I needed to deal with all of my different meshes and for cheering me on. I am also grateful that she provided me with the FORTRAN code for the Variable Neighbourhood Search (VNS).

A sincere thank you goes to my friend Dr. Ahmad Alkandari for making sure the English in this thesis reads smoothly and for providing many important tips and suggestions, and also for always being there when I needed him. As luck would have it, we wrote our thesis side by side, which I very much enjoyed.

I am grateful to my wife Safiah, who supported and encouraged me to complete my thesis. I also dedicate this work to my children Alaa, Shaimaa, Dalya, and

Nasser. I am also grateful to my brothers, sisters, and my friends for their support and prayers.

Author's Publications

1). M'Hallah R., Alkandari A., and Mladenovic N., Packing Unit Spheres into the Smallest Sphere Using VNS and NLP, Computers & Operations Research 40 (2013) 603-615

2). M'Hallah R., and Alkandari A., Packing Unit Spheres into a Cube Using VNS, Electronic Notes in Discrete Mathematics 39 (2012) 201-208

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Background

The optimization of non-linear problem is a classical situation that is frequently encountered in nature. In most cases, finding the global optimum for these mathematical programs is difficult. This is due to the complexity of the topography of the search space and the exorbitant by high computational costs of the existing approaches. Despite the advancement of computational technologies, the computational costs remain excessively high. An alternative to these expensive methods are heuristic approaches which provide good quality solutions in reasonable computational time. There are several classes of heuristic methods. They can be grouped as local search and global search, or as nature-inspired and non-nature-inspired, or as single-start and multiple-start (or population based). The search can itself be a steepest descent/ascent or more elaborate, temporarily accepting, non-improving solutions; or of a prohibiting nature. A heuristic is successful if it balances the intensification and diversification of the search within the neighborhoods of the search space.

A relatively new heuristic that has proven successful is the variable neighborhood search (VNS). The VNS searches for a (near-) global optimum starting from several initial solutions, and changes the size or structure of the neighborhood of the current local optimum whenever its search stagnates. In other words, it opts for an exploration phase every time its exploitation search fails in improving its current incumbent.

Another option in the search for a global optimizer is hybrid heuristics. These heuristics target overcoming limitations in terms of intensification and diversification through hybridization. For instance, genetic algorithms are known for their diversification whereas simulated annealing and tabu search are notorious for their intensification. Thus, their hybridization has resulted in the resolution of many complex combinatorial optimization problems.

In this thesis a particular non-linear program is addressed using hybrid heuristics inspired from the variable neighborhood search framework. Specifically, it considers the problem of packing three-dimensional unit spheres into three-dimensional containers, where the objective is to minimize the size of the container.

1.2 Problem description

This thesis considers packing n identical spheres, of radius one, without overlap into the smallest containing sphere S. This problem, is a three-dimensional variant of the Open Dimension Problem: all small items (which are spheres) have to be packed into a larger containers (which should be a sphere or a cube) and the size of the container has to be minimized. The problem is equivalent to finding the coordinates

(x_i, y_i, z_i) of every sphere $i \in I = 1, ..., n$, and the dimensions of the container such that every sphere $i \in I$ is completely contained within the object and no pair (i, j) of spheres overlap.

1.3 Motivation

The sphere packing problem, which consists of packing spheres into the smallest sphere or cube, has many important real-life applications including materials science, radio surgical treatment, communication, and other vital fields. In materials science, random sphere packing is a model for the structure of liquids, proteins, and glassy materials. The model is used in the study of phenomena such as electrical conductivity, fluid flow, stress distribution and other mechanical properties of granular media, living cells, random media chemistry and physics. The model is also applied in the investigation of processes such as sedimentation, compaction and sintering. In radio surgical treatment planning, sphere packing is crucial to X-ray tomography. In digital communication and storage, it emerges in the packing of compact disks, cell phones, and internet cables. Other applications of sphere packing are encountered in powder metallurgy for three-dimensional laser cutting, in the arranging and loading of containers for transportation, in the cutting different natural by formed crystals, in the layout of computers, buildings, etc. Sphere packing is an optimization problem, but it is debatable whether it should be classified as continuous or discrete. The positions of the spheres are continuous whereas the structure of an optimal configuration is discrete. A successful solution technique should tackle these two aspects simultaneously.

1.4 Contribution

Packing unit spheres into three-dimensional shapes is a non-convex optimization problem. It is NP hard (Non-deterministic Polynomial-time hard), since it is an extension of packing unit circles into the smallest two-dimensional shape, which is, in turn, NP hard [32]. Thus, the search for an exact local extremum is time consuming without any guarantee of a sufficiently good convergence to an optimum. Indeed, the problem is challenging. As the number of unit spheres increases, identifying a reasonably good solution becomes extremely difficult. In addition, the problem has an infinite number of solutions with identical minimal radii. In fact, any solution may be rotated or reflected or may have free spheres which can be slightly moved without enlarging the radius of the container sphere. Finally, there is the issue of computational accuracy and numerical precision. Solving the problem via non-linear programming solvers is generally not successful. Most solvers are not geared towards identifying the global optima. Subsequently, the problem should be solved by a mixture of search heuristics with local exhaustive (exact) searches of the local minima or their approximations. This thesis follows this line of research.

This thesis models the problems as non-linear programs and approximately solves them using a hybrid heuristic which couples a variable neighborhood search (VNS) with a local search (LS). VNS serves as the diversification mechanism whereas LS acts as the intensification one. VNS investigates the neighborhood of a feasible local minimum (u) in search of the global minimum, where neighboring solutions are obtained by shaking one or more spheres of (u) and the size of the neighborhood is varied by changing the number of shaken spheres, and the distance and the direction each sphere is moved. LS intensifies the search around a solution (u) by subjecting its neighbors to a sequential quadratic algorithm with a non-monotone line search (as the NLP solver).

The results and findings extracted from this research are beneficial both to academia and industry. The application of the proposed approach to other problems will allow solving larger instances of other difficult problems. Similarly, its application in an industrial setting will greatly reduce industrial waste when packing spherical objects; thus, reducing the costs, not only in its monetary aspects but also in its polluting aspect. VNS can be applied to other problems with high industrial relevance such as vehicle routing, facility location and allocation, and transportation. Thus, this research can contribute into the mainstream applications of economic and market-oriented packing strategies.

1.5 Outline

A tutorial and a detailed survey on VNS methods and applications, including the following is presented in chapter 2 of this thesis:

- ∗ a general framework for the principles of VNS.
- ∗ the principles of VNS.
- ∗ three earlier heuristic approaches.
- ∗ different VNS algorithms.
- ∗ four parallel implementations of VNS.

The pseudo code, strengths and limitations of each VNS heuristic, and some of its successful applications areas are also discussed in section 2. In Chapter 3 adopts a VNS algorithm is adopted to pack unit spheres into the smallest three-dimensional sphere, and the most prominent literature on the subject is reviewed. The proposed hybrid approach proposed, and its relative and absolute performance is detailed. The superiority of the proposed approach in terms of numerical precision is demonstrated, provide new upper bounds for 29 instances, and showing the utility of the local and variable neighborhood search. The effects of varying the VNS parameters is also presented. In Chapter 4 the problem of packing unit spheres into a cube is discussed, along with a detailed up-to-date literature review on the problem. A description of the solution, highlighting its symmetry reduction is presented along with the formulation augmenting techniques. The results presented compared are then to existing upper bounds. The results obtained matches 16 existing upper bounds and are accurate to 10^{-7} for the others. Finally, in Chapter 5 the thesis, is summarized future directions of research are recommended, and other applications of VNS' based hybrid heuristics are proposed.

Chapter 2

Variable Neighborhood Search

2.1 Introduction

The variable neighborhood search (VNS) is a meta-heuristic or a framework for building heuristics. The VNS has been widely applied during the past two decades because of its simplicity. Its essential concept consists in altering neighborhoods in the quest for a (near-) global optimal solution. In the VNS is investigated the search space are researched via a descent search technique, in which immediate neighborhoods are searched; then, deliberately or at irregular intervals, a more progressive search is adopted where in neighborhoods that are inaccessible from its current point are investigated. Regardless of the type of search, one or a few focal points within the current neighborhood serve as starting points for the neighborhood search. Thus, the search bounces from its current local solution to a new one, the search and only if it discovers a preferred solution, or undertakes a predefined number of successive searches without improvement. Hence, the VNS is not a trajectory emulating technique (as Simulated Annealing or Tabu Search) and does not define prohibited moves as in Tabu Search.

An optimization problem can be defined as identifying the best value of a real-valued function f over a feasible domain set X. The solution x where $x \in X$ is feasible if it satisfies all the constraints for some particular problem. The feasible solution x^* is optimal (or is a global optimum) if it yields the best value of the objective function f among all $x \in X$. For instance, when the objective is to minimize f, the following holds:

$$f(x^*) = \min\{f(x) : x \in X\} \tag{2.1}$$

i.e., $x^* \in X$ and $f(x^*) \leq f(x), \forall x \in X$. The problem is combinatorial if the solution space X is (partially) discrete. A neighborhood structure $N(x) \subseteq X$ of a solution $x \in X$ is a predefined subset of X. The solution $x' \in N(x)$ is a neighbor of x. It is a local optimum of equation (2.1) with respect to (w.r.t.) the neighborhood structure $N(x)$ if $f(x') \leq f(x)$, $\forall x \in N(x)$. Accordingly, any steepest descent method (i.e., technique that just moves to a best neighbor from the current solution) is trapped when it reaches a local minimum.

To escape from this local optimum, meta-heuristics, or general frameworks for constructing heuristics, adopt some jumping procedures which consist of changing the focal point (or incumbent solution) of the search, or accepting deteriorating moves, or accepting prohibiting moves, etc. The most widely known among these techniques are Genetic Algorithms (GA), Simulated Annealing (SA) and Tabu Search (TS) as detailed in Reeves [58] and Glover and Kochenberger [19].

A brief overview of the VNS and its different variants is presented in this chapter. The most attention will be paid to parallel VNS techniques. In section 2.2 background information is provided on generic VNS and its principles. Two approaches that are predecessors of VNS are also presented. These approaches are the variable metric approach and iterated local search. In section 2.3 the different versions of VNS are detailed, along with the pseudo code, some applications, and evidence of their strengths for each version. In section 2.4 the parallelization of VNS four approaches are presented. Finally, In section 2.5 a summary is presented.

2.2 Preliminaries

The VNS adopts the strategy of escaping from local minima in a systematic way. It deliberately updates the incumbent solution in search of better local optima and in escaping from valleys [44, 45, 25, 27, 28]. VNS applies a sophisticated system to reach a local minimum; then investigates a sequence of diverse predefined neighborhoods to increase the chances that the local minimum is the global one. Specifically, it fully exploits the present neighborhood by applying a steepest local descent starting from different local minima. When its intensification stops at a local minimum, VNS jumps from the revamped local minimum into a different neighborhood whose structure is different from that of the present one. In fact, it diversifies its search in a pre-planned manner unlike in SA and TS, which permit non-improving moves within the same neighborhood or temper with the solution path. This systematic steepest descent within different neighborhood structures led to the VNS frequently outperforming other meta-heuristics while providing pertinent knowledge about the problem behavior and characteristics; thus, permitting the user to improve the design of the VNS heuristic both in terms of solution quality and runtime while preserving the simplicity of the implementation.

The simplicity and success of the VNS can be attributed to the following three observations:

Observation 1 A local minimum with respect to one neighborhood structure is not necessarily so for another neighborhood.

Observation 2 A global minimum is a local minimum with respect to all possible neighborhood structures.

Observation 3 For many problems, local minima with respect to one or several neighborhoods are relatively close to each other.

Differently stated, the VNS stipulates that a local optimum frequently provides some useful information on how to reach the global one. In many instances, the local and global optima share the same values of many variables.

However, it is hard to predict which ones these variables are. Subsequently, the VNS undertakes a composed investigation of the neighborhoods of this local optimum until an improving one is discovered.

The section 2.2 presents two fundamental variable neighborhood approaches proposed in a different context than the VNS. In section 2.2.1 the variable metric procedure, originally intended for unconstrained continuous optimization problems, where using different metrics is synonym to different neighborhoods is discussed. In section 2.2.2 the iterated local search, also known as fixed neighborhood search, intended for discrete optimization problems is detailed.

2.2.1 The variable metric procedure

This procedure emanates from gradient-based approaches in unconstrained optimization problems. These approaches consist of taking the largest step in the best direction of descent of the objective function at the current point. Initially, the search space is an n-dimensional sphere. Adopting a variable metric modifies the search space into an ellipsoid. The variable metric consists of a linear transformation. This procedure was applied to approximate the inverse of the Hessian of a positive definite matrix within n iterations.

2.2.2 Iterated local search (LS)

The most basic form of VNS is the Fixed Neighborhood Search (FNS) [34], also known as the Iterated local search (ILS) [23]. In this strategy an initial solution $x \in X$ is chosen, and subjected to a local search $Local - Search$ (x), and x^* is declared as the best current solution. Then, iteratively undertaken the following three steps are: First, the current best solution x^* is perturbed obtaining a solution $x' \in X$ by using the procedure $\underline{Shake}$ (x^*). Second, the procedure $Local - Search$ (x') is applied to the perturbed solution x' to obtain the local minimum x''. Third and last, the current best solution x^* is updated if x'' is a better local minimum. This three-step iterative approach is repeated until a stopping criterion is met. The stopping condition can be set as the maximal runtime of the algorithm or the maximal number of iterations or an optimality gap of the objective function if a good bound is available. However, this latter condition is rarely applied while the number of maximal iterations without improvement of the current solution is the most widespread criterion. Table 2.1 gives the pseudo code of FNS.

The perturbation of x^*, via procedure $\underline{Shake}(x^*)$, is not neighborhood dependent; thus the adoption of the term “fixed”. The size of the perturbation should not be too small or too large [23]. In the former case, the search will stagnate at x^* since the perturbed solutions $x' \in X$ will be very similar to the starting point x^* (i.e., within the immediate neighborhood of x^*). In the latter case, the search will be assimilated to a random restart of the local search thus hindering the biased

Table 2.1: Pseudo Code of the FNS

```
1   Find an initial solution $x \in X$
2   $x^* \leftarrow Local - Search\ (x)$
3   Do While (Stopping condition is $False$):
  3.1   $x' \leftarrow \underline{Shake}(x^*)$
  3.2   $x'' \leftarrow Local - Search(x')$
  3.3   If $f(x'') < f(x^*)$ set $x^* = x''$
```

sampling of FNS where the sampled x' solutions are obtained from a fixed neighborhood whose focal point is x^*. The perturbation is to allow jumps from valleys and should not be easily undone. This is the case of the 4-opt (known also as the double-bridge) for the traveling salesman problem where the local search initiated from x' yield good quality local minima even when x^* is of very good quality [23]. The more different the sampled solution x' is from the starting point x^*, the stronger (and most likely the more effective) the perturbation is, but in such a case the cost of the local search is [23] more.

The procedure $\underline{Shake}(x^*)$ may take into account the historical behavior of the process by prohibiting certain moves for a number of iterations (as in Tabu Search) or by taking into account some precedence constraints, or by fixing the values of certain variables, or by restricting the perturbation to only a subset of the variables.

The procedure $Local - Search(x')$ is not necessarily a steepest descent type search. It is any optimization approach that does not reverse the perturbation of the $\underline{Shake}$ procedure while being reasonably fast and yielding good quality solutions, as is the case in Tabu Search, simulated annealing, a two-opt exhaustive search, etc.

In general, it is advantageous to have a local search that yields very good quality solutions.

Finally, the update of the biased sampling focal point (step 3.3 in Table 2.1) is not necessarily of a steepest-descent type. It may be stochastic; for example, the focal point of the search moves to a non-improving local minimum in search of a global optimum. This strategy resembles the acceptance criterion of simulated annealing. It may also be a combination of a steepest descent and of a stochastic search, which tolerates small deteriorating moves with a certain probability. The choice of the best updating approach should weigh the need and usefulness of the diversification versus intensification of the search [23].

Chiarandini and Stutzle [14] applied FNS (referring to it as ILS) to the graph-coloring problem, using the solution obtained by a well-known algorithm as the initial solution, Tabu Search as the local search, and two neighborhood structures with the first being a one-opt and the second, taking into consideration all pairs of conflicting vertices. They adopt a special array structure to store the effect of their moves and speed the search. Their ILS improves the results of many benchmark instances from the literature. Grosso et al.[9] employ FNS to identify max-min Latin hypercube designs. Their problem consists in scattering n points in a k-dimensional discrete space such that the positions of the n points are distinct. The objective is to maximize the distance between every pair of points. For their ILS, they generate the initial solution randomly but suggest that using a constructive heuristic may yield better results. They apply a local search that swaps the first component of two critical points, and a perturbation mechanism that extends the local search to larger neighborhoods. Their ILS improves some existing designs. It is as good as a multi-start search but faster. It is better than the latest implementation of simulated annealing to the problem and competes well with the periodic design approach.

Hurtgen and Maun [33] successfully applied FNS in positioning synchronized phasor-measuring devices in a power system network. They identified the best-known solution for benchmark instances. When minimizing the total cost (expressed in terms of the number of phasor-mearument units to be installed) while ensuring full coverage of the network, they improved the best-known feasible solution by as much as 20%. Burke et al. [18] compare the performance of ILS to that of six variants of hyper-heuristics a cross three problem domains. A variant combines one of the two heuristic selection mechanisms (uniformly random selection of a heuristic from a prefixed set of heuristics and reinforcement learning with Tabu Search) and one of three acceptance criteria (naive acceptance, adaptive acceptance, and great deluge). Even though it was not consistently the best approach for all tested instances of one-dimensional bin packing, permutation flow shop, and personnel scheduling problems, ILS outperforms, overall, the six variants of the hyper-heuristics. The authors stipulate that the robustness of ILS is due to its successful balancing of its intensification and diversification strategies, to its simple implementation which requires no parameter, and to the proximity of the local minima for the three classes of problems.

2.3 Elementary VNS algorithms

Because of its simplicity and successful implementations, VNS has gained wide applicability. This has resulted in several variants of VNS. A successful design takes into account the application area, the problem type, and nature of the variables, not to mention the runtime. Sections 2.3.1 to 2.3.6 present some mostly used VNS versions: the variable neighborhood descent (VND), the reduced variable neighborhood search (RVNS), the basic variable neighborhood search (BVNS), the general variable neighborhood search (GVNS), the skewed variable neighborhood search (SVNS) and the variable neighborhood decomposition search (VNDS). In section 2.3.7 the distinctive characteristics of each of these variants are highlighted.

2.3.1 The variable neighborhood descent

VND is an objective form of VNS. It is founded on the first and third observations, which stipulate that a local optimum for a given neighborhood $N_k(x)$, $k \in K = \{1 \ldots, \bar{k}\}$ is not necessarily a local optimum for a second neighborhood $N_k(x)$, $k' \in K$ and $k' \neq k$ and that a global optimum is optimal over all $\bar{k}$ neighborhoods. The parameter $\bar{k}$ is the maximum number of neighborhoods. Therefore, it investigates a neighborhood $N_k(x), k \in K$ to obtain a local optimum and searches within other neighborhoods for an improving one. It returns to the initial neighborhood structure every time it identifies an improving solution while it moves to a more distant neighborhood every time it fails to improve the current solution. It stops when the current solution is optimum for all $\bar{k}$ neighborhoods. Table 2.2 gives a detailed algorithm of VND.

The VND is particularly useful for large-sized instances of combinatorial optimization problems where the application of local search meta-heuristics tend to use large CPU times (central processing unit). Nikolic et al. [48] implemented a VND for the covering design problem where neighboring solutions are built by adding and removing subsets to the current solution and resulted in the improving of 13 upper bounds.

Hansen et al.[50] consider berth allocation problem, i.e., the order in which ships will be allocated to berths. They minimize the total berthing costs where the total cost includes the costs of waiting and handling of ships as well as earliness

Table 2.2: Pseudo Code of the VND

```
1   Find an initial solution $x \in X$, set the best solution $x^*$=$x$, and $k$=1.
2   Do While ($k \leq \bar{k}$):
  2.1  Find the best neighbor $x' \in N_k(x)$ of $x$
  2.2  If $f(x') < f(x)$ then
         set $x = x'$ and $k = 1$;
         If $f(x') < f(x^*)$, set $x^* = x'$
         else
         set $k = k + 1$.
```

or tardiness. They propose a VNS, and compare it to a multi-start heuristic, a genetic algorithm, and a memetic algorithm. They show that the VNS is on average the better of the three approaches. Qu et al. [57] apply the VND to the delay-constrained least-cost multicast routing problem, which reduces to the NP-complete delay-constrained Steiner Tree. They investigate the effect of the initial solution by considering two initial solutions: one based on the shortest path and one on bounded-delay, and consider three types of neighborhoods. They show that the VND outperforms the best existing algorithms. Kratica et al. [35] use a VND within large shaking neighborhoods to solve a balanced location problem. This latter problem consists of choosing the location of p facilities while balancing the number of clients assigned to each location. They show that their implementation is very fast and reaches the optimal solution for small instances. For large instances where the optimal solution value is unknown, the VNS outperforms a state-of-the-art modified genetic algorithm.

2.3.2 The reduced variable neighborhood descent

The RVNS is a slightly different provision of the VNS; yet, it retains its basic structure. It is based upon the third principle of the VNS which states that a global optimum is the best solution a cross all possible neighborhoods. It is a stochastic search. The outer loop constitutes the stopping. The inner loop conducts a search over a fixed set of $\bar{k}$ nested neighborhoods $N_1(x) \subset N_2(x) \subset \ldots, \subset N_{\bar{k}}(x)$ that are centered on a randomly generated focal point x. A solution $x' \in X$ is generated using a stochastic procedure $\underline{Shake}$ (x, k) which slightly perturbs the solution x. For instance, in non-linear programming where the final solution of any optimization solver depends on the solution it is fed, procedure $\underline{Shake}$ (x, k) would alter the value of one or more of the variables of x by small amounts δ where the altered variable(s) and δ are randomly chosen. Subsequently, either the focal point of the search is changed to the current solution x or an enlarged neighborhood search is to be undertaken. Specifically, when $f(x') < f(x)$, the inner loop centers its future search space on the most reduced neighborhood around x' (i.e., it sets $x{=}x'$ and $k{=}1$). Otherwise, it enlarges its sampling space search by incrementing k but maintains its current focal point x. Every iteration of the inner loop can be assimilated to injecting a cut to the minimization problem, if each new bound improves the existing one; thus, to fathoming a part of the search space.

Table 2.3 gives a detailed pseudo code of the RVNS. This approach is well suited for multi-start VNS-based strategies where RVNS can be replicated with different initial solutions $x \in X$ and the best solution over all replications is retained. It behaves like a Monte Carlo simulation except that its choices are systematic rather than random. It is a more general case of Billiard simulation; a technique that was applied to point scattering within a square.

Table 2.3: Pseudo Code of the RVNS

```
1   Find an initial solution $x \in X$; set the best solution $x^*=x$, and $k$=1.
2   Choose a stopping condition.
3   Do While (Stopping condition is $False$):
        Do While ($k \leq \bar{k}$)
    3.1  $x' \leftarrow \underline{Shake}\ (x, k)$
    3.2  If $f(x') < f(x)$, then
             set $x=x'$ and $k$=1;
             If $f(x') < f(x^*)$ set $x^*=x'$
             else
             set $k=k+1$.
```

RVNS is particularly useful when the instances are large or when the local search is expensive [53, 47]. Computational proof is given in section 2.3.4 for the high school timetabling problem, where the performance of the RVNS is been to better than be the GVNS, which applies simulated annealing as the local search [59]. Hansen et al. [53] further substantiated the claim regarding the speed of the RVNS. They compare the performance of the RVNS to that of the fast interchange heuristic for the p-median problem. They show that the speed ratio can be as large as 20 for comparable average solutions. For the same problem, Crainic et al. [72] provide additional performance analysis of a parallel implementation of the RVNS. Maric et al. [43] hybridize the RVNS with a standard VNS. They compare the performance of the resulting heuristic to a swarm optimization algorithm and a simulated annealing one for the bi-level facility location problem with clients' preference, but unlimited capacity for each facility. They show that the hybrid heuristic outperforms the other two in large-scale instances and is competitive in the case of smaller instances.

2.3.3 The basic variable neighborhood search

The BVNS is a descent, first-improvement method [53]. It is a hybridization of the VND and the RVNS. It evokes variable neighborhoods at irregular intervals but consistently applies a steepest descent to a (near-) global optimum. It is basically an RVNS where the inner loop applies to the solution x' obtained via $\underline{Shake}(x, k)$ a steepest descent procedure $Local - Search(x', k)$, which searches around the solution x' for the best first improving solution $x'' \in N_k(x')$, where $k \in K$. In fact, it accelerates the search by opting for the first rather than the best improving solution. The BVNS adopts the same stopping criteria as the RVNS; that is, the inner loop stops if the investigation of $\bar{k}$ successive neighborhoods centered on x yield no improvement, whereas the outer loop stops when a user-defined stopping criterion is satisfied. This condition is generally related to total runtime or the number of iterations without improvement of the best solution. Table 2.4 provides a detailed description of the BVNS.

The BVNS is well suited to multi-start VNS-based strategies, where a local search is applied to perturbed solutions. Toksari and Guner [73] provide computational proof that the local search of the BVNS is more efficient than the VND in the case of unconstrained optimization problems. They consider the case of a non-convex but differentiable function with many local minima but a single global minimum. Their VNS applies a standard descent heuristic with the directions of descent randomly generated. Their results are competitive with existing approaches when tested on existing benchmark instances. Sanchez-Oro and Duarte [62] show that the BVNS is superior to the RVNS and the VND where finding near-optima for both the min-max and min-sum variants of the vertex-cut minimization problems for short and long time horizons. M'Hallah and Alkandari [55] and M'Hallah et

Table 2.4: Pseudo Code of the BVNS

```
1    Find an initial solution $x \in X$; set the best solution $x^*$=$x$, and $k$=1.
2    Choose a stopping condition.
3    Do While (Stopping condition is $False$):
         Do While ($k \leq \bar{k}$)
3.1  $x' \leftarrow \underline{Shake}(x, k)$.
3.2  $x'' \leftarrow Local - Search\ (x', k)$.
3.3  If $f(x'') < f(x)$, then
         set $x = x''$ and $k = 1$;
         If $f(x'') < f(x^*)$ set $x^* = x''$
         else
         set $k = k + 1$.
```

al. [56] apply BVNS for packing unit spheres into the smallest cube and sphere, respectively, highlighting the utility of the local search.

2.3.4 The general variable neighborhood search

The GVNS is a low-level hybridization of the BVNS with the RVNS and the VND [51, 52]). A detailed description of the GVNS is given in Table 2.5. First, it applies the RVNS to obtain a feasible initial solution to the problem (step 2 in Table 2.5) in lieu of directly sampling a feasible solution $x \in X$ (as in step 1 in Table 2.4). This is particularly useful in instances where finding a feasible solution is in itself an NP-hard problem. Second, it replaces the local search (Step 3.2 in Table 2.4) of the BVNS with a VND (Step 5.2 in Table 2.5). In addition, it samples the points x' from the k^{th} neighborhood $N_k(x)$ of the current solution x. In fact, it uses procedure

$\underline{Shake}(x, k)$.

Table 2.5: Pseudo Code of the GVNS

```
1   Find an initial solution $x$.
2   $x' \leftarrow RVNS\ (x)$ starting with $x$ to obtain a feasible solution $x' \in X$ .
3   Set the best solution $x^*=x'$ the current solution $x=x'$ and the neighbor-
    hood counter $k=1$.
4   Choose a stopping condition.
5   Do While (Stopping condition is $False$)
       Do While ($k \leq \bar{k}$):
  5.1   $x' \leftarrow \underline{Shake}\ (x, k)$ .
  5.2   $x'' \leftarrow VND\ (x')$.
  5.3   If $f(x'') < f(x)$ , then
          set $x=x''$ and $k=1$;
          If $f(x'') < f(x^*)$ set $x^*=x''$
          else
          set $k=k+1$.
```

GVNS was applied to large-sized vehicle routing problems with time windows [64, 46] and to the traveling salesman problem [46]. The implementation of Mladenovic et al. [46] provides the best upper bounds in more than half of the existing benchmark instances. It was particularly useful because identifying initial feasible solutions and maintaining feasibility during the shaking procedure and the neighborhood investigation was a challenging task.

2.3.5 The skewed variable neighborhood search

The SVNS is a modified BVNS where a non-improving solution x'' may become the new focal point of the search, when this solution x'' is far enough from the current one x, but its value $f(x'')$ is not much worse than $f(x)$, the value of the current solution x [27]. The SVNS is motivated by the topology of search spaces. The search gets trapped in a local optimum and cannot leave it because all neighboring solutions are worse. Yet, if it opts for a "not-too-close" neighborhood, it may reach the global optimum. This neighborhood should not be too far and the neighbor should not be too much worse can to enable to return to the current neighborhood, if the exploration fails to identify an improving solution.

Table 2.6: Pseudo Code of the SVNS

```
1    Find an initial solution $x \in X$; set the best solution $x^*=x$, and $k=1$.
2    Choose a stopping condition.
3    Do While (Stopping condition is $False$)
         Do While ($k \leq \bar{k}$):
  3.1    $x' \leftarrow \underline{Shake}\ (x, k)$.
  3.2    $x'' \leftarrow Local - Search\ (x', k)$.
  3.3    If $f(x'') - \alpha\rho(x, x'') < f(x)$, then
          set $x=x''$ and $k=1$;
          If $f(x'') < f(x^*)$ set $x^*=x''$
          else
          set $k=k+1$.
```

The difference between x and x'', is measured in terms of a distance $\rho(x, x'')$ while the difference between $f(x'')$ and $f(x)$ is considered tolerable if it is less than an expression pondering the distance $\rho(x, x'')$ by a parameter α. That is, the condition $f(x'') < f(x)$ of step 3.3 in Table 2.4 is replaced by $f(x'') - \alpha\rho(x, x'') < f(x)$. A detailed description of the BVNS is provided in Table 2.6.

The utility of the SVNS is demonstrated by Hansen and Mladenovic [27] for the weighted maximum satisfiability of logic problem for which the SVNS performs better than the GRASP (Greedy Randomized Adaptive Search Procedure) and Tabu Search, for medium and large problems, respectively. The choice of α is generally based on an analysis of the behavior of a multiple start VNS.

2.3.6 The variable neighborhood decomposition search

The VNDS is used in the particular case, where the set X of feasible solutions is finite. Yet, its extension to the infinite case is possible. It is one of two techniques designed to reduce the computational time of local search algorithms. Even though they investigate a single neighborhood structure, local search heuristics tend to have their runtime increase significantly as the size of the combinatorial problems become large [53]. It is "a BVNS within successive approximations decomposition" scheme [53]. That is, both the VNDS and the BVNS have the same algorithmic steps except that procedures $Shake$ and $Local - Search$ of steps 3.1 and 3.2 of Table 2.4, respectively, are implemented on a partial solution y of free variables, whereas all other variables remain fixed as in x throughout the random selection of $x' \in X$ and the local search for an improving solution $x'' \in X$. A detailed description of the pseudo-code of the VNDS is given in Table 2.7.

Obtaining a random solution $x' \in X$ from the current solution $x \in X$ via procedure $\underline{Shake}(x, k, y)$ entails choosing values for the partial solution y, which consists of a set of free variables of x, while ensuring that the new value of every

Table 2.7: Pseudo Code of the VNDS

```
1    Find an initial solution $x \in X$; set the best solution $x^*$=$x$, and $k$=1.
2    Choose a stopping condition.
3    Do While (Stopping condition is $False$)
         Do While ($k \leq \bar{k}$):
  3.1    $x' \leftarrow \underline{Shake}\ (x, k, y)$.
  3.2    $x'' \leftarrow Local - Search\ (x', y, k)$.
  3.3    If $f(x'') < f(x)$ , then
          set $x$=$x''$ and $k$=1;
          If $f(x'') < f(x^*)$ set $x^*$=$x''$
          else
          set $k$=$k$+1.
```

free variable is different from its current value in x. The choice of the free and fixed variables constitutes the heart of the decomposition approach. It can follow some rule of thumb or some logical pattern. Note that the cardinality of y set is equal to $n - k$.

Obtaining a local optimum $x'' \in X$ from the current solution $x' \in X$ via procedure $Local - Search(x', y, k)$ entails finding the best values of the partial solution y, given that the all other fixed variables of x' retain their values in the local optimum x''. It is equivalent to undertaking a local search on the reduced search space of y. It is possible not to apply a local search and to simply implement an inspection approach or exactly solve the reduced problem if the number of fixed variables is very large.

The VNDS is useful in binary integer programming in general [36], and solving large-scale p-median problems [53] in particular. The p-median problem involves choosing the location of p facilities among m potential ones in order to satisfy the demand of a set of users at least cost. Lazic et al. [36] provide computational proof that the VNDS performs best on all performance measures of solution approaches including computation time, optimality gap, and solution quality. Hanafi et al. [60] tackle the 0-1 mixed-integer programming problem using a special VNDS variant. This latter exploits the information obtained from an iterative relaxation-based heuristic in its search. This information serves to reduce the search space and avoid the reassessment of the same solution during different replications of the VNDS. The heuristic adds pseudo-cuts based on the objective function value to the original problem to improve the lower and upper bounds; thus, it reducing the optimality gaps. The approach yields the best average optimality gap and running time for binary multi-dimensional knapsack benchmark instances. It is inferred that the approach can yield tight lower bounds for large instances.

2.3.7 Comparison of the VNS variants

The VNS was designed for combinatorial problems, but is applicable to any global optimization problem. It explores distant neighborhoods of incumbent solutions in search for global optima. It is simple and requires very few parameters. The main characteristics of its seven variants are summarized in Table 2.8. The VNS has been hybridized at different levels with other heuristics and meta-heuristics. For instance, Kandavanam et al. [20] consider route multi-class network communication planning problem in order to satisfy service quality. They hybridize the VNS with a genetic algorithm and apply the hybrid heuristic to maximizing the residual bandwidth of all links in the network, while meeting the requirements of the expected quality of service.

Table 2.8: Main Characteristics of the VNS Variants

VND	deterministic change of neighborhoods; more likely to reach a global minimum, and if many neighborhood structures are used.
RVNS	useful for very large instances, for which local search is costly; best with $\overline{k} = 2$, and analogous to a Monte-Carlo simulation, but more systematic.
BVNS	deterministic and stochastic changes of neighborhoods, and systematic change of neighborhoods.
GVNS	VND is used as a local search within the BVNS; very effective, and useful for low-level hybridization.
SVNS	useful for flat problems, and useful for clustered local optima.
VNDS	a two-level VNS (decomposition at the first level), and useful for large instances.

2.4 Parallel VNS

Most sequential heuristic approaches are being implemented as parallel approaches. The increasing tendency towards parallelism is motivated both by the potential reduction of computational time (through the segmentation of the sequential program, and by the expansion of the investigation of the search space (through the provision of more processors and memory for the computing device). The VNS is among the sequential heuristics that were implemented in a parallel computing environment. Four parallelization techniques have been so far proposed [29]. The first two techniques are basic: the parallelization of the neighborhood local search and of the VNS itself by assigning the same VNS algorithm to each thread, and retaining the best solution among all solutions reported by the threads. There is no cooperation among the individual threads. The remaining two techniques on the other hand utilize cooperation mechanisms to upgrade the performance level of the algorithm. They are more complex and involve intricate parallelization [39, 72] to synchronize communication. A detailed description of these four techniques follows.

The first parallelization technique is the synchronous parallel VNS (SPVNS). It is the most primary parallelization technique [39]. It is designed to shorten the

Table 2.9: Pseudo Code of the SPVNS

```
1   Find an initial solution $x \in X$; set the best solution $x^*$=$x$, and $k$=1.
2   Choose a stopping condition.
3   Do While (Stopping condition is $False$)    Do While ($k \leq \bar{k}$):
  3.1  $x' \leftarrow \underline{Shake}\ (x)$.
  3.2  Divide the neighborhood $N_k(x')$ into $n_p$ subsets.
  3.3  For every processor $p$, $p$=1...,$n_p$, $x''_p \leftarrow Local - Search\ (x', k)$.
  3.4  Set $x''$ such that $f(x'') = \max_{p=1,n_p} \{f(x''_p)\}$
  3.5  If $f(x'') < f(x)$, then
         set $x$=$x''$ and $k$=1;
         If $f(x'') < f(x^*)$ set $x^*$=$x''$
         else
         set $k$=$k$+1.
```

runtime through the parallelization of the local search of the sequential VNS. In fact, the local search is the most time-demanding part of the algorithm. The SPVNS splits the neighborhood into n_p parts and assigns each subset of the neighborhood to an independent processor, which returns to the master processor an improving neighbor within its subset of the search space. The master processor sets the best among the n_p neighbors returned by the n_p processors as the current solution. Table

2.9 provides the pseudo code of the SPVNS, adapted from Garcia et al. [39].

The second parallelization technique is the reproduced parallel VNS (RVNS) or replicated parallel VNS or simply a multi-start VNS. It consists of n_p parallel independent searches, where n_p is the number of parallel threads of the computing device. Each independent search operates an independent VNS on a separate processor. Table 2.10 provides a detailed description of the RPVNS. It can be perceived as a multi-start RVNS where the best solution n_p is retained as the best solution except that, the n_p replications are undertaken in parallel instead of sequentially [39].

Table 2.10: Pseudo Code of the RPVNS

```
1   Choose a stopping condition.
2   Do While ($p \leq n_p$)
 2.1   Find an initial solution $x_p \in X$; set the best solution $x_p^*$=$x_p$, and
       $k$=1.
 2.2   Do While (Stopping condition is $False$)
       Do While ($k \leq \bar{k}$)
        • $x' \leftarrow \underline{Shake}(x_p)$.
        • $x'' \leftarrow Local - Search(x', k)$.
        • If $f(x'') < f(x)$, then
              set $x = x''$ and $k = 1$;
              If $f(x'') < f(x_p^*)$ set $x_p^* = x''$;
              else
              set $k = k + 1$.
3– Set $x^*$ such that $f(x^*) = \max_{p=1,n_p} \{f(x_p^*)\}$.
```

The third parallelization technique is the replicated shaking VNS (RSVNS) proposed by Garcia et al. [39]. It applies a synchronous cooperation mechanism through a conventional master-slave methodology. The master processor operates a sequential VNS, and sends its best incumbent to every slave processor, which shakes this solution to obtain a starting point for its own local search. In turn, each slave returns its best solution to the master. This latter compares all the solutions obtained by the slaves, retains the best one and subjects it to its sequential VNS. This information exchange is repeated until a stopping criterion is satisfied. The

Table 2.11: Pseudo Code of the RSVNS

```
1   Find an initial solution $x \in X$; set the best solution $x^*$=$x$, and $k$=1.
2   Choose a stopping condition.
3   Do While (Stopping condition is $False$)
        Do While ($k \leq \bar{k}$)
  3.1   For $p$=1...,$n_p$
          • Set $x_p$=$x$
          • $x' \leftarrow$ Shake ($x'_p$).
          • $x''_p \leftarrow Local - Search$ ($x'_p, k$).
  3.2   Set $x''$ such that $f(x'') = \max_{p=1,n_p} \{f(x''_p)\}$
  3.3   If $f(x'') < f(x)$, then
          set $x$=$x''$ and $k$=1;
          If $f(x'') < f(x^*_p)$ set $x^*_p$=$x''$;
          else
          set $k$=$k$+1.
```

fact that VNS permits changing neighborhoods and types of local search makes this type of parallelization possible. Different neighborhoods or local searches can be performed by independent processors and the resulting information is channeled to the master processor which analyzes its results to obtain the best solution and conveys it to the other slaves. Maintaining a trace of the last step undertaken by every processor strengthens the search as it avoids the duplication of computational efforts. Table 2.11 provides a detailed description of the RSVNS. It can be perceived as a VNS with a multi-start shaking and local search where the best local solution among n_p ones is retained as the best local solution and routed to each of the n_p processors [39]. A comparison of the performance of these first three parallel VNS algorithms is undertaken by Garcia et al. [39] for the p-median problem.

The fourth and last parallelization technique is the cooperative neighborhood VNS (CNVNS) suggested by Crainic et al. [72] and Moreno-Perez et al. [38]. It is particularly suited to combinatorial problems such as the p-median problem. It deploys a cooperative multi-search strategy to the VNS while exploring a central-memory mechanism. It coordinates many independent VNSs by asynchronously exchanging the best incumbent obtained so far by all processors. Its implementation preserves the simplicity of the original sequential VNS ideas. Yet, its asynchronous cooperative multi-search offers a broader exploration of the search space thanks to the different VNSs being applied by the different processors. Each processor undertakes an RVNS, and communicates with a central memory or the master, every time it improves the best global solution. In turn, the master relays this information to all other processors so that they update their knowledge about the best current solution. That is, no information is exchanged in terms of the VNS itself. The master is responsible for launching and stopping the parallel VNS. In case of Crainic et al. [72], the parallel algorithm is an RVNS where the local search is omitted; resulting in a faster algorithm. Table 2.12 and Table 2.13 gives a detailed description of the CNVNS as originally intended and as described by Moreno-Perez et al. [38]. A comparison of the performance of the four parallel VNS algorithms is undertaken by Moreno-Perez et al. [38] for the p-median problem.

Table 2.12: Pseudo Code of the CNVNS: Master's Algorithm

```
1   Find an initial solution $x \in X$; set the best solution $x^* = x$, and $k_p$=1,
    $p = 1, \ldots, n_p$
2   Choose a stopping condition.
3   Set $x_p = x$
4   For $p = 1, \ldots, n_p$ , launch RVNS with $x_p$ as its initial solution.
5   When processor $p'$ sends $x''_{p'}$.
  5.1   If $f(x''_{p'}) < f(x)$, then update current best solution setting $x = x''_{p'}$
        and send it to all $n_p$ processors.
6 When processor $p'$ requests best current solution, send $x$
```

Table 2.13: Pseudo Code of the CNVNS: Slave's Algorithm

```
1   Obtain initial solution $x_p$ from master; set the best solution $x^*_p = x_p$,
    and randomly choose $k \in \{1, \ldots, \bar{k}\}$
2   Do While $(k \leq \bar{k})$
  2.1  $x' \leftarrow$ Shake $(x)$.
  2.2  If $f(x'_p) < f(x_p)$, then
          set $x_p = x'_p$ and $k$=1;
          if $f(x'_p) < f(x^*_p)$ set $x^*_p = x_p$.
          else
          send $x^*_p$ to the master;
          receive $x_p$ from master;
          set $k$=$k$+1.
```

2.5 Discussion and conclusion

The VNS is a general framework for solving optimization problems. Its successful application to different continuous and discrete problems is advocating for its wider use to non-traditional areas such as neural networks and artificial intelligence. The VNS owes its success to its simplicity and to its limited number of parameters:the stopping criterion and maximal number of neighborhoods. Depending on the specific problem at hand, a VNS variant may be deemed more appropriate than other variants. In fact, a judicious choice of the neighborhood structure and local search strategy could determine the success of an approach. The local search may vary from exact optimization techniques for relaxed or reduced problems to gradient descent, line search, steepest descent, to meta-heuristics like Tabu Search and simulated annealing.

Chapter 3

Packing Unit Spheres into the Smallest Sphere Using the VNS and NLP

3.1 Introduction

Sphere packing problems (SPP) consist comprise of packing spheres into a sphere of the smallest possible radius. It has many important real-life applications including materials science, radio surgical treatment, communication, and other vital fields. In materials science, random sphere packing is a model to represent the structure of liquids, proteins, and glassy materials. The model is used to study phenomena such as electrical conductivity, fluid flow, stress distribution and other mechanical properties of granular media, living cells, random media chemistry and physics. SPP also entails the investigation of processes such as sedimentation, compaction, and sintering [76]. In radio surgical treatment planning, sphere packing is crucial to

X-ray tomography. In [75] a problem on how to pack minimum number of unequal spheres into a three dimensions bounded region, in connection with radio-surgical treatment planning, is studied. It is used for treating brain and sinus tumor (see Figure 3.1).

Figure 3.1: X-Ray

During the operation, medical unit should not effect other organs. Gamma knife is one of more effective radio-surgical modalities. It can be described as radioactive dosage treatment planning. A high radiation dose is called the *shot* which can be viewed as a sphere. These *shots* are ideal equal spheres to be packed into a container, but also they could be of different spheres. The tumor can be viewed as an approximate spherical container. No *shots* are allowed outside the container, which means that the radiation *shots* are only hitting the tumor cells and not the healthy ones. Multiple *shots* are usually applied at various locations and may touch the boundary of the target. They avoid overlapping and touch each other. The stronger the high packing density, the more doses delivered. The target of the search to minimize the number of *shots* into the container (The tumor). As a result, this approach has met with certain success in medical applications. Dynamic programming algorithm is being used to find the optimum number of *shots*.

In digital communication and storage, it emerges in compact disks, cell phones, and the Internet [15, 74]. The most frequent application of minimum sphere packing problem is connected with location of antennas. For example it is crucial in antenna location in some large warehouse or container yards (see Figure 3.2). Each article usually has radio frequency identifier (RFID) or tag. The management of the warehouse wants to locate minimum number of antennas that cover all warehouse such that vehicles, connected with the antenna system, are able to find any article. The radii of each antenna are known in advance. This system provides real-time location visibility, illuminating vital information that is needed [6].

Figure 3.2: Warehouse

Other applications of sphere packing are encountered in powder metallurgy for three-dimensional laser cutting [42]; cutting different natural crystals; layout of computers, buildings, etc.

In this chapter the special case of packing unit spheres into the smallest sphere (PSS) is considered. PSS entails packing n identical spheres, of radius 1 unit, without overlap into the smallest containing sphere S. The goal is to search for the best packing of the n spheres into S, where the best packing minimizes waste. According to the Typology of Cutting and Packing of Wascher et al. [79], PSS are a three-dimensional variant of the Open Dimension Problem since all small items (which are spheres) have to be packed and the dimension of the large object (which is a sphere) is not given, and has to be minimized. PSS is equivalent to finding the coordinates (x_i, y_i, z_i) of every sphere $i,\ i = 1, \ldots, n$, the radius r and coordinates (x, y, z) of S, such that no pair of spheres $(i, j) \in I \times I$ and $i < j$ overlap. Formally, the problem can be stated as finding the optimal level of the decision variables $r,\ (x, y, z)$, and $(x_i, y_i, z_i),\ i = 1, \ldots, n$, such that

$$\min \quad r \tag{3.1}$$

$$\text{subject to} \quad (x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2 \leq (r-1)^2 \quad i = 1, \ldots, n, \tag{3.2}$$

$$(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 \geq 4 \quad 1 \leq i < j \leq n, \tag{3.3}$$

$$r \geq \underline{r}, \tag{3.4}$$

where $\underline{r}$ is a strictly positive lower bound on r. The first set of constraints enforces the containment of every sphere within S. The second set enforces the no overlap constraint of any pair of distinct spheres. Finally, the last constraint provides a positive lower bound for the radius of the containing sphere.

PSS is a non-convex optimization problem [32, 75]. It is NP-hard [75], since it is an extension of packing unit circles into the smallest circle, which is in turn NP-hard [32]. Thus, the search for an exact local extremum is time-consuming, without any guarantee of a sufficiently good convergence to the optimum [65]. Indeed, PSS is a challenging problem. As the number of unit spheres increases, identifying a reasonably good solution becomes extremely difficult [32]. In addition, PSS has an infinite number of solutions with identical minimal radii. In fact, any solution may be rotated or reflected or may have free spheres which can be slightly moved without enlarging the radius of the containing sphere. Finally, there is the issue of computational accuracy and precision.

Solving PSS via an off-the-shelf non-linear programming (NLP) solver is generally not successful. In fact, most of these solvers fail to identify global optima because of PSS' difficulty, which gets further amplified as the problem size increases (since the number of local minima increases too). Subsequently, PSS should be solved by a mixture of search heuristics with local exhaustive (exact) search of local minima or their approximations as suggested in [67, 32]. This chapter follows this line of research. It tackles PSS using a hybrid heuristic which combines these two main components. It applies a variable neighborhood search (VNS) which investigates different neighborhoods of the incumbent solution, and a special purpose NLP solver to locally search for one or more minima within each neighborhood.

Section 3.2 reviews the most prominent literature in the area of sphere packing. Section 3.3 details the proposed approach. Section 3.4 investigates its performance. Specifically, this section demonstrates the superiority of the proposed approach in terms of accuracy and precision, provides new upper bounds for 14 instances, and determines the utility of VNS and the local search. Finally, section 3.5 is a summary.

3.2 Literature review

Sphere packing is considered when the container three-dimensional into which the sphere are to be packed is a cube, a parallelepiped, a cylinder, a polytope, or a sphere.

Gensane [22] adapts the Billiard simulation to the problem of identifying the largest radius of n **identical** spheres that fit inside a unitary **cube**. Billiard simulation is a stochastic method that mimics the idealized movement of billiard balls inside a domain, with the initial centers of the balls and their directions being randomly fixed. The resulting configuration emanates from probabilistic choices. To improve the convergence of the stochastic algorithm, different types of local searches were considered. These included moving the spheres randomly in a random walk simulation, decreasing the magnitude of the stochastic movements as the size of the spheres increased, and perturbing all spheres simultaneously.

Stoyan and Yaskov [66] focus on finding the minimal height of a **parallelepiped** (L, W, H) that packs n **identical** spheres. They model the problem as a NLP with a linear objective function and linear and quadratic constraints. They approximately solve the model using a special search tree in conjunction with a modification of the Zoutendijk method [80] of feasible directions to calculate local minima, and a modification of the decremental neighborhood method to search for an approximation to the global minimum.

Using the same technique, Stoyan and Yaskov [68] tackle the problem of minimizing the height H of a right circular **cylinder**, of known radius r, that packs n **identical** spheres. The authors obtain the best results to date for $n = 498, 499$, and 500. Their approach is very effective for $n \leq 500$, and can handle instances for $n \leq 2000$.

Stoyan et al. [77] use techniques similar to those considered in [67] to identify a packing of n **non-identical** spheres, each of radius $r_i,\ i \in I$, into a **parallelepiped** of fixed length L and width W but of variable height H with the objective of minimizing H. The authors provide numerical results with up to 60 spheres.

Yaskov et al. [21] tackle the problem of maximizing the number of **identical** spheres that can be packed into a **cylindrical** composed domain. They construct a mathematical model based on the concept of Φ-functions [78], and design a solution algorithm based on a modification of the optimization method by groups of variables.

Wang [75] formulates mathematically the automated radio surgical treatment planning problem as the packing of spheres into a three-dimensional region with a packing density greater than a given threshold level. He proves that this packing problem is NP-complete and proposes an approximate algorithm to solve it.

Sutou and Dai [70] assimilate the automated radio surgical treatment planning problem to packing non-identical spheres in a three-dimensional polytope with the objective of maximizing the volume of the packed spheres. They formulate the problem as a non-convex optimization one with quadratic constraints and a linear objective function. They propose a variety of algorithms which outperform the generic algorithm for the general non-convex quadratic program. In fact, they incorporate heuristics into the generic algorithm to strengthen its efficiency. They demonstrate its efficiency computationally for limited problem sizes.

Stoyan and Yaskov [69] consider the problem of packing the maximal number of congruent hyper spheres of unit radius into a larger hyper sphere of given radius. They construct a mathematical model, and approximately resolve it by solving a sequence of packing subproblems –whose objective functions are linear– using the Zoutendijk feasible direction method. Starting points are generated in accordance with the lattice packing of hyperspheres translated on various vectors or in a random way. They improve the convergence of their sequential approach by perturbing the lattice packing.

Hales [24] proves the Kepler conjecture which stipulates that the packing density of identical three-dimensional spheres cannot exceed $\frac{\pi}{\sqrt{18}}$. Bezdek [11] brings forth a strong Kepler conjecture which alleges that the density of at least two equal spheres in a sphere of the three-dimensional space of constant curvature is less than $\frac{\pi}{\sqrt{18}}$.

Birgin and Sobral [13] propose a reduced model for PSS. They aggregate the $\frac{n(n-1)}{2}$ non-overlap constraints into a single constraint:

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \max\{0, 4 - (x_i - x_j)^2 - (y_i - y_j)^2 - (z_i - z_j)^2\} = 0, \tag{3.5}$$

and implement a reduction approach that avoids computing many of the $\frac{n(n-1)}{2}$ terms of the sum. In fact, only terms relative to spheres that could potentially be touching are included. They solve the resulting model, starting from several initial solutions, using a local solver that is based on an augmented Lagrangian method for smooth general constrained minimization. They report solutions for instances with up to 100 spheres.

Liu et al. [37] adopt an unconstrained model for the problem where they aggregate all the constraints of PSS and "Lagrange" them into the objective function:

$$\begin{aligned} \min \quad & r + \sum_{i=1}^{n} d_{i0}^2 + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij}^2, && (3.6)\\ \text{where} \quad & d_{ij}^2 = \max\{0, 4 - (x_i - x_j)^2 - (y_i - y_j)^2 - (z_i - z_j)^2\}, && (3.7)\\ \text{and} \quad & d_{i0}^2 = \max\{0, (x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2 - (r-1)^2\}. && (3.8) \end{aligned}$$

Their model is inspired from quasi physics where spheres that are in contact have extrusive elastic energy, and a system is in equilibrium when all elastic energies are nil. They solve the resulting model using a heuristic which combines the energy landscape paving (ELP) method with an adaptive step length gradient descent procedure. ELP is an optimization strategy that mimics the Monte Carlo simulation while using concepts from simulated annealing.

Pfoertner [1] posts the best known radii r_i^* of n identical spheres that can be packed into a containing sphere of radius one. This problem can be stated as: equal sphere packing (EPP)

$$\max \quad r_i \tag{3.9}$$

$$\text{Subject to} \quad (x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2 \le (1 - r_i)^2 \quad i = 1, \ldots, n, \tag{3.10}$$

$$(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 \ge 4r_i^2 \quad 1 \le i < j \le n, \tag{3.11}$$

$$0 \le r_i \le 1 \quad i = 1, \ldots, n. \tag{3.12}$$

The EPP model is equivalent model to the PSS model. In fact, its solution value r_i^* can be mathematically mapped to the PSS optimal solution value r^* :

$$r^* = \tfrac{1}{r_i^*}. \tag{3.13}$$

3.3 Proposed approach

PSS is a difficult problem. It has an infinite number of alternative optima, an exponential number of local optima which are not globally optimal, and an uncountable set of stationary points – i.e., solutions that satisfy the Karush Kuhn Tucker (KKT) conditions but which do not correspond to local optima. Moreover, the solution to a PSS problem is sensitive to the choice of the initial solution (as is the case of non-linear optimization). Finally, the larger the number of spheres to be packed, the larger the number of local minima and stationary points. Thus, the simple multi-start global optimization strategies need more local minimizations to reach a global minima [13].

PSS is herein solved using an approach that applies Schittkowski local search (LS) as the intensification strategy and variable neighborhood search (VNS) as the diversification strategy. In sections 3.3.1 and 3.3.2, detail LS and VNS, respectively, are detailed.

3.3.1 Schittkowski's local search

An approximate solution to PSS can be obtained using an off-the-shelf NLP solver. However, its quality (i.e., its deviation from the optimal radius) depends both on the initial solution fed to the solver and on its proximity to the global optimum. In addition, there is no guarantee that the obtained solution is a local optimum. In fact, the solution can be a saddle point. Moreover, the solver can stop for various reasons such as no improving search direction, errors in evaluating function and gradient values, violation of regularity conditions, etc.

To overcome the shortcomings of NLP solvers, a PSS problem is approximately solved via a better adapted NLP solver. It applies a NonLinear Programming with Non-Monotone and Distributed Line Search (NLPQLP), the sequential quadratic programming (SQP) algorithm to a non-monotone line search of Schittkowski [63]. NLPQLP is designed to resolve smooth NLP problems. When the solver aborts because of computational errors caused by inaccurate function or gradient evaluations, a non-monotone line search is activated. Internal restarts are performed in case of errors when computing the search direction due to inaccurate derivatives. Additional automated initial and periodic scaling with restarts is implemented. Schittkowski [63] reports that the extensive computational investigation of the code on benchmark instances clearly demonstrated the improvements brought up by the non-monotone search. He further indicates that more than 90 % of the 306 tested instances, were successfully solved, subject to a stopping tolerance of 10^{-7}.

NLPQLP requires an initial solution and bounds for the variables of the problem. Providing loose bounds on the variables may hinder the convergence of the solver to a feasible direction or induce computational errors. The radius r can be bounded in two ways. A straightforward lower bound $\underline{r}$ assumes that the spheres are not solid, and that S contains no unused volume. Subsequently, the minimal volume of S is the sum of the volumes of the n spheres. It follows that the radius of this utopian sphere is $\underline{r} = \sqrt[3]{n}$. A tighter lower bound uses the Kepler conjecture [24] that the density of the best packing of $n \geq 2$ non-overlapping spheres cannot exceed $\frac{\pi}{\sqrt{18}}$. Consequently, $\underline{r} = \sqrt[3]{\frac{\pi}{\sqrt{18}} n}$.

The upper bound $\bar{r}$ is estimated by assuming that the spheres can be fitted into a cube of side $c = 2\lceil \sqrt[3]{n} \rceil$, where $\lceil a \rceil$ is a integer such that $\lceil a \rceil - 1 < a \leq \lceil a \rceil$. The sphere fitting this cube has a radius $\bar{r} = \sqrt{3(\frac{c}{2})^2}$.

From a mathematical point of view, having S centered at $(0,0,0)$ or $(\bar{r},\bar{r},\bar{r})$ yield equivalent configurations since they can be obtained from each other via a simple transformation. However, LS did not handle negative values of the coordinates will and often failed to converge to a feasible solution in the former case. Thus, each of the coordinates $(x_i,\ y_i,\ z_i)$ of a small sphere $i,\ i = 1, \ldots, n$, are bounded by 1 and $2\bar{r} - 1$. Similarly, these bounds apply to the coordinates $(x,\ y,\ z)$ of the containing sphere S. Tighter bounds can be used, but in many instances it let to infeasible solutions. Our experimental investigation further indicated that most NLP solvers (including LS) face difficulties in restoring containment feasibility if the initial r fed to the solver is too small (even in the absence of overlap). Very loose bounds are counter-productive too. They cause the local search to converge to local optima that are too far from the global optimum.

It is believed that feeding the solver with a feasible initial solution speeds up the search process and provides a better estimate for the global minima. Meanwhile many newer NLP approaches claim their robustness with respect to the initial solution and its feasibility. Herein, LS is started from both a feasible solution and randomly generated solutions.

When started from an initial feasible solution, the search is denoted SLS. The feasible solution is obtained using a very simple constructive heuristic which packs the spheres in layers in a cube of side c. It positions sphere $i,\ i=1,\dots,n$, in (x_i, y_i, z_i), where $x_i = 2d+1,\ y_i = 2b+1,\ z_i = 2a+1$, with a, b, d integers satisfying $i = ac^2 + bc + d$. This heuristic is obviously very simple and does not yield a compact packing of the spheres. On the other hand, it is very fast, causes no truncation or roundoff errors, offers the spheres enough degrees of freedom to move, and gives LS a feasible descent direction. Evidently, the lattice configuration is another feasible configuration. However, it was discarded since it is suspected to correspond to the optimal packing in many instances and represents a local minima for the others. Thus, it may hinder the search for better quality optima.

When started with a random initial solution (which is not necessarily feasible), the search is denoted as RLS. The random configuration is generated by arbitrarily selecting the coordinates (x, y, z) and $(x_i, y_i, z_i),\ i = 1, \dots, n$, from the uniform$[1, 2\bar{r} - 1]$, and setting $r = \bar{r}$. When RLS is restarted k_M times, the search is denoted as M-RLS (for Multistar RLS).

3.3.2 Variable neighborhood search

VNS is a metaheuristic which exploits systematically the idea of neighborhood change, both in descent to local minima and in escape from the valleys which contain them [27, 26, 28, 54, 45]. Its success emanates from three main characteristics. First, a local minimum with respect to one neighborhood is not necessarily a local minimum for another neighborhood. Second, a global minimum must be a local minimum with respect to all possible neighborhood structures. Third, in many problems including PSS, local minima with respect to one or more neighborhoods are relatively close to each other.

In VNS the search is initiated with a feasible solution $\mathbf{u}$ for PSS with a radius $r_{\mathbf{u}}$. It sets $k = 1$, and randomly generates a solution $\mathbf{u}'$ from a neighborhood $N_k(\mathbf{u})$ of $\mathbf{u}$. It then applies a local search to $\mathbf{u}'$ (SLS or RLS depending on whether $\mathbf{u}'$ is feasible or not) to identify a local minimum $\mathbf{u}" \in N_k(\mathbf{u})$. If $r_{\mathbf{u}"}$ improves the current solution, the best solution is updated, i.e, the current solution $\mathbf{u}$ is set to $\mathbf{u}"$. That is, if $r_{\mathbf{u}"} < r_{\mathbf{u}^*}$, then $r_{\mathbf{u}^*}$ is reset to $r_{\mathbf{u}"}$, and $\mathbf{u}^*$ to $\mathbf{u}"$. Differently stated, the focus of the search is re-centered on $\mathbf{u}"$. Consequently, k is reset to 1 and the search is restarted by generating a random solution from the neighborhood $N_1(\mathbf{u})$. On the other hand, if $\mathbf{u}"$ does not improve the current solution $r_{\mathbf{u}}$, a different neighborhood of $\mathbf{u}$ is investigated. Specifically, k is incremented, and a different random solution $\mathbf{u}'$ from the new neighborhood $N_k(\mathbf{u})$ of $\mathbf{u}$ is obtained, and the local search is applied to $\mathbf{u}'$ to get $\mathbf{u}"$. It represents the inner loop of the method. When it is completed, then k is set to 1 and the inner loop is started again. It is repeated until the VNS runtime is smaller than the maximum runtime allowed. The detailed algorithm of VNS is given in Algorithm 1.

A solution $\mathbf{u}'$ from a neighborhood $N_k(\mathbf{u})$, $k = 1, \ldots, \bar{k}$, of $\mathbf{u}$ is generated

Algorithm 1 Detailed Algorithm of the VNS for PSS

```
Input
  1  An initial feasible solution $\mathbf{u} = (x_i, y_i, z_i)_{i=1,\ldots,n}$, and $r_{\mathbf{u}}$ the corresponding
     radius of $S$.
  2  $\bar{t}$, the maximum runtime for VNS.
  3  $\bar{k}$, the maximum number of neighborhoods.
Output
  1  An approximate optimal solution $\mathbf{u}^* = (x_i^*, y_i^*, z_i^*)_{i=1,\ldots,n}$, and its associated
     radius $r_{\mathbf{u}^*}$.
Algorithm
  1  Set $\mathbf{u}^* = \mathbf{u}$ and $r_{\mathbf{u}^*} = r_{\mathbf{u}}$.
  2  Do while $t \leq T$
    2.1  Set the neighborhood type $k = 1$.
    2.2  Do while $k \leq \bar{k}$
      2.2.1  Apply the procedure Shake$(k, \mathbf{u}, \mathbf{u}')$ to obtain a random solution $\mathbf{u}' \in$
             $N_k(\mathbf{u})$, where $N_k(\mathbf{u})$ is the $k^{\text{th}}$ neighborhood of $\mathbf{u}$.
      2.2.2  Apply the local search with $\mathbf{u}'$ as the starting solution, to get the new
             solution $\mathbf{u}''$ and its associated value $r_{\mathbf{u}''}$.
      2.2.3  If $r_{\mathbf{u}''} < r_{\mathbf{u}^*}$, set $k = 1$, $\mathbf{u}^* = \mathbf{u}''$, $r_{\mathbf{u}^*} = r_{\mathbf{u}}$, and $\mathbf{u} = \mathbf{u}''$; else, set $k = k+1$.
```

using procedure <u>*Shake*</u>$(k, \mathbf{u}, \mathbf{u}')$. Let $S_{[k]}$ denote the k^{th} closest sphere to the center of the containing sphere S, and $d_{[k]}$ the distance that separates the centers of $S_{[k]}$ and S. For $k \leq n$, procedure <u>*Shake*</u>$(k, \mathbf{u}, \mathbf{u}')$ translates sphere $S_{[k]}$ by $\delta^x_{[k]}, \delta^y_{[k]}$ and $\delta^z_{[k]}$ in the x, y and z directions, respectively. The translation distances are randomly generated from the Uniform$[-\Delta, \Delta]$, where Δ is the neighborhood parameter. If this translation results in the violation of the lower (resp. upper) bound of a variable, this latter is reset to its lower (resp. upper) bound. For $n < k \leq 2n$, procedure

$\underline{Shake}(k, \mathbf{u}, \mathbf{u}')$ translates every sphere $S_{[k']}$, $k' = 1, \ldots, k-n$, by $\delta^x_{[k']}, \delta^y_{[k']}$ and $\delta^z_{[k']}$ in the x, y and z directions, respectively. Finally, for $k > 2n$, procedure $\underline{Shake}(k, \mathbf{u}, \mathbf{u}')$ translates every sphere $S_{[k']}$, $k' = 1, \ldots, k - 2n$, as long as its $d_{[k']} \leq 3.0$. This strategy reflects the fact that smaller variations on the positions of spheres closest to the center of S have a more sizeable impact on the final radius of S than do variations on spheres closer to its surface.

The VNS can only be started from a feasible solution $\mathbf{u}$. Herein, a distinction is made between the cases where this feasible solution is obtained from SLS and from RLS, and the two corresponding searches are denoted as VNS-SLS and VNS-RLS. Finally, we consider a final version of the search, denoted M-VNS-RLS, where VNS-RLS is restarted with the k_M solutions obtained by M-RLS.

From a pure mathematical point of view, PSS can be solved by setting either (x, y, z) or any (x_i, y_i, z_i), $i \in \{1, \ldots, n\}$, to $(0, 0, 0)$ or to any other triplet value of interest. In fact, any solution can be translated in a way that results in one of the spheres been centered at the triplet of interest. However, our extensive computational investigation showed that this is not always advisable. Even though this strategy is supposed to reduce the size of the problem and to alleviate some of the difficulties caused by symmetry, it also limits, in many instances, the capacity of the NLP solver to converge to a good quality local optimum. Subsequently, two versions of the proposed hybrid heuristic are considered: without a fixing position (denoted hereafter with a suffix N) and with a fixing position (denoted hereafter with a suffix F).

Our computational investigation showed that fixing the three coordinates of a sphere is not judicious in most instances. Therefore, it was decided that the fixing should be limited to only one coordinate of one of the spheres. Because of the special structure of the problem, there is no guideline to the choice of the coordinate to be fixed. Consequently, when fixing is adopted, procedure $\underline{Shake}(k, \mathbf{u}, \mathbf{u}')$ fixes $x_{[1]}$ if $k \leq n$, $y_{[1]}$ if $n < k \leq 2n$, and $z_{[1]}$ if $k > 2n$. The coordinate is fixed at its corresponding value in $\mathbf{u}$ whatever that value is. It is obviously not 0. Since there is no guarantee that using the VNS with fixing yields better results than without fixing, this search option can be used as a diversification strategy.

Subsequently, VNS is run with three types of diversification strategies: fixing versus not fixing a coordinate of one of the spheres, varying the search time T of the VNS, and varying the neighborhood size Δ. The first strategy, as explained in the previous paragraph, is motivated by the fact that fixing one of the variables would reduce the search space by eliminating a large number of equivalent solutions (such as symmetrical ones, or ones obtained by rotation or reflection, etc.). The last two strategies are motivated by the fact that an enlarged neighborhood, or an intensive search within a neighborhood, could lead to a very good quality local minimum that is better than one obtained with smaller values or a less intensive search. However, this local minimum may cause the search to stagnate and prohibit its escape toward a better quality local minimum.

3.4 Computational results

The objective of the computational investigation is threefold: to show that the results obtained using the proposed approach are more accurate and precise than those in the literature; to provide new upper bounds for 14 instances; and to investigate the effect of the three diversification strategies, and point out the utility of the VNS and of the multi-start on the performance of LS.

All codes were written in `Fortran` using double precision computation. The Schittkowski code was modified for two reasons: the stopping tolerance is 10^{-14}, and it is automatically restarted with a new random solution if its current run does not yield a feasible solution that satisfies the optimality conditions or if the current solution is not the best solution encountered during the search. Unless otherwise stated, the VNS heuristics use $\Delta = 1, \bar{k} = 50$, and $\bar{T} = 12$ seconds. All instances were run on a Pentium IV dotted with a 3.20 GHz processor and 4 GB of memory. The results are discussed in the following section.

3.4.1 Overall performance

Table 3.1 displays the best-known and best-obtained radii. Column 1 indicates the problem size n, where $n = 1, \ldots, 50$. Columns 2 to 5 display the best reported radii $\hat{r}_B,\ \hat{r}_0,\ \hat{r}_1,\ \hat{r}_2$, where $\hat{r}_B,\ \hat{r}_0,\ \hat{r}_1,\ \hat{r}_2$ are obtained from [13, 1, 2, 3], respectively. In Columns 3 to 5, for a given n, the minimum of $\hat{r}_0,\ \hat{r}_1$, and $\hat{r}_2$ is highlighted. Finally, Columns 6 to 10 report r_{H}, the best radius r obtained by heuristic H, $\text{H} \in \mathcal{H} = \{\text{SLS,VNS-SLS, M-RLS, VNS-RLS, M-VNS-RLS}\}$, and the $\min\limits_{\text{H}\in\mathcal{H}}\{r_{\text{H}}\}$ is highlighted. For the VNS heuristics, the reported radii are the best over all runs: F and N, $T = 6$ and 12 seconds, and $\Delta = 0.1, 0.2, 0.5, 0.75, 1.0, 1.25, 1.5$. For M-RLS and M-VNS-RLS, the reported radii are obtained using the same set of $k_M = 50$ initial random solutions. In fact, for a given $\Delta,\ T,\ \bar{k}$, and neighborhood structure, $r_{\text{SLS}} \geq r_{\text{VNS-SLS}},\ r_{\text{M-RLS}} \geq r_{\text{M-VNS-RLS}}$, and $r_{\text{VNS-RLS}} \geq r_{\text{M-VNS-RLS}}$. Some of these relationships may not hold in the values in Table 3.1 since the displayed results are summarized over all $\Delta,\ T$ and neighborhood structures.

The analysis of Table 3.1 involves the discussion of issues related to accuracy, precision, and convergence. The results of Birgin and Sobral [13] presented in Column 2 of Table 3.1 are most likely inaccurate. The authors admitted the inaccuracy of their results in the two-dimensional case and proposed improved re-

Table 3.1: Best Local Minima

n	$\hat{r}_B$	$\hat{r}_0$	$\hat{r}_1$	$\hat{r}_2$	r_{SLS}	$r_{\text{VNS-SLS}}$	$r_{\text{M-RLS}}$	$r_{\text{VNS-RLS}}$	$r_{\text{M-VNS-RLS}}$
1	1.0000005143								
2	1.9999202433				2.00000000000000	2.00000000000000	2.00000000000000	2.00000000000000	2.00000000000000
3	2.1546210616	2.1547006	2.1547006		2.15470053824456	2.15470053824456	2.15470053837444	2.15470053837425	2.15470053837425
4	2.2246870768	2.2247450	2.2247451		2.41421356237309	2.22474487138673	2.22474487138940	2.22474487138675	2.22474487138670
5	2.4141267898	2.4142139	2.4142139		2.41421356237298	2.41421356236821	2.41421356237147	2.41421356236832	2.41421356236814
6	2.4141694573	2.4142139	2.4142139		2.52752523165194	2.41421356237177	2.41421356237309	2.41421356237130	2.41421356236983
7	2.5911991946	2.5912543	2.5912536		2.61368668160301	2.61368668160195	2.61368668160301	2.61368668160200	2.61368668159921
8	2.6452822222	2.6453291	2.6453291		2.73205080757142	2.73205080756874	2.73205080756888	2.73205080756887	2.73205080756886
9	2.7320034623	2.7320516	2.7320508		2.74182565152031	2.73205080756797	2.73205080756887	2.73205080756831	2.73205080756539
10	2.8324167450	2.8324648	2.8324648		2.83246456105392	2.83246456105390	2.83246456105391	2.83246456105391	2.83246456105391
11	2.9019556947	2.9021140	2.9021132		2.90211303259061	2.90211303259030	2.90211303259030	2.90211303259030	2.90211303259013
12	2.9020105031	2.9021140	2.9021132		2.99941691897225	2.90211303259025	2.90211303259030	2.90211303259030	2.90211303258981
13	2.9997977416	3.0000003	3.0000003		3.10221395734388	2.99999999999943	2.99999999999997	2.99999999999945	2.99999999999936
14	3.0910230631	3.0911462	3.0911462		3.11144281851829	3.09114544488870	3.09114544488871	3.09114544488870	3.09114544488861
15	3.1415101582	3.1416438	3.1416429	3.1416409	3.32152692721521	3.14164262494870	3.14164262494871	3.14164262494871	3.14164262494867
16	3.2155339532	3.2156843	3.2156833	3.2156822	3.22035762742487	3.21568303201004	3.21568303201004	3.21568303201004	3.21568303201004
17	3.2711271196	3.2712462	3.2712451	3.2712451	3.31921100101264	3.29680836874176	3.29684043850944	3.29680836874176	3.29680836874176
18	3.3188360708	3.3189898	3.3189898	3.3189953	3.43690158448033	3.38084090740455	3.38084090740455	3.38084090740455	3.38084090740455
19	3.3858513801	3.3860164	3.3860164	3.3860198	3.55861664021641	3.46435524031985	3.46116202704004	3.46435524031985	3.46116202704004
20	3.4733633429	3.4735405	3.4735405	3.4740195	3.59742989177187	3.56548622298350	3.57187266979232	3.56548622298350	3.56548622298350
21	3.4862092764	3.4863528	3.4863528	3.4863492	3.62033409540338	3.60153022804217	3.60153022804217	3.60153022804217	3.60153022804217
22	3.5796881818	3.5800655	3.5800655	3.5799437	3.68953464150661	3.64051666844219	3.64051666844219	3.64051666844219	3.64051666844219
23	3.6273889908	3.6275165	3.6275165	3.6352929	3.76703739927985	3.76703739927985	3.76804140699504	3.76654989681805	3.76654989681805
24	3.6852536176	3.6853955	3.6853955	3.6854675	4.22574928847579	3.83281214836974	3.83289514992629	3.83281214836974	3.83281214836974
25	3.6872896999	3.6874285	3.6874271	3.6884037	4.04050812837879	4.04050812837879	3.68839179615006	3.75327834408109	3.68839179615006
26	3.7471886302	3.7474293	3.7474054	3.7498238	4.39215002079962	4.39080038824415	3.74740577652750	3.74740577652750	3.74740577652750
27	3.8132974881	3.8134171	3.8134157	3.8150465	4.46410161513775	4.46410161513775	3.81615120368277	3.84660056076369	3.81615120368277
28	3.8415149358	3.8417093	3.8416399	3.8447343	4.05290053391191	3.84164027814771	3.84177917884272	3.84164027814771	3.84164027814771
29	3.8769556364	4.0334762	3.8770885	3.8786901	3.90588517026782	3.87708910315835	3.87747914930252	3.87708910315835	3.87708910315835
30	3.9163640724	3.9165002	3.9164910	3.9249549	3.93900318427918	3.91649166155428	3.91654074598313	3.91649166155428	3.91649166155428
31	3.9506163857	3.9507874	3.9507546	3.9507734	4.01340436656033	3.95075448490565	3.95136931193641	3.95075448490565	3.95075448490565
32	3.9873139176	3.9874412	3.9874412	3.9886404	4.27254890158653	3.98744038931492	3.99772008339166	3.98744038931492	3.98744038931492
33	4.0197789249	4.0207406	4.0199001	4.0208602	4.08580036058113	4.01990091595852	4.01994758165121	4.01990091595852	4.01990091595852
34	4.0475629785	4.0478914	4.0477194	4.0484847	4.10040899215648	4.04771997123058	4.04944723850525	4.04771997123058	4.04771997123058
35	4.0842793950	4.0844935	4.0844050	4.0854179	4.13002114560296	4.08440574075337	4.08444990930670	4.08440574075337	4.08440574075337
36	4.1128778440	4.1129904	4.1129887	4.1358717	4.30716585797470	4.11298932968653	4.11298932968653	4.11298932968653	4.11298932968653
37	4.1546179855	4.1559012	4.1547805	4.1590245	4.24424648243793	4.15478125199120	4.15658310328267	4.15478125199120	4.15478125199120
38	4.1575074608	4.1577500	4.1577345	4.1604260	4.23715102793316	4.15766926004822	4.15773497480818	4.15766926004822	4.15766926004822
39	4.2238096708	4.2273985	4.2239505	4.2247035	4.25333789139973	4.23907241802319	4.25486395699313	4.22394975626618	4.22394975626618
40	4.2551873286	4.2557610	4.2553336	4.2576563	4.38558965541323	4.27145569016750	4.29321652173874	4.26365269398190	4.26365269398190
41	4.2961778465	4.2980380	4.2963465	4.2973417	4.31673932506242	4.31499920733667	4.32646345045588	4.31409218937400	4.30805958902068
42	4.3080119020	4.3132916	4.3081351	4.3081351	4.37739382983185	4.34481244093924	4.36786365788228	4.35355395885825	4.33740094016591
43	4.3528504653	4.3603289	4.3528792	4.3541301	4.39892791988379	4.37938295264594	4.40770331022630	4.38261922568558	4.37711338192368
44	4.3827055183	4.3829161	4.3828335	4.3843688	4.63248403341084	4.43037520199247	4.43643887928122	4.40321999967785	4.40321999967785
45	4.4068820245	4.4103145	4.4070026	4.4131795	4.49709897273965	4.46784626453004	4.46170130820936	4.45527185482123	4.44323242599026
46	4.4409860772	4.4421523	4.4411264	4.4412073	4.54196325546583	4.50305842563616	4.50219963456297	4.47847363222525	4.47847363222525
47	4.4739825106	4.4746333	4.4741328	4.4774382	4.59256828247835	4.52534868057796	4.54766625099119	4.54502187645631	4.51745560770452
48	4.4961576629	4.4973331	4.4962836	4.4971511	4.74031835202101	4.57213881540195	4.60308683627023	4.57449980224184	4.55711696552466
49	4.5191083524	4.5265746	4.5192613	4.5252964	4.83383651985818	4.61638412473422	4.66024661415626	4.66004684175310	4.61507723897088
50	4.5504157703	4.5589573	4.5505657		4.76300934968712	4.69158717299905	4.74017473378151	4.70703758917657	4.66922650552785

sults for circle packing [12]. Because they used the same algorithm for packing unit spheres, their results are probably erroneous. For example, when packing two unit spheres into the smallest sphere, the authors obtain $r_B = 1.9999202433$ while the correct exact answer should be 2.0000000000. Similarly, when packing 13 circles, $r_B = 2.9997977416$, while the correct answer should be 3.0000000000. Thus, no comparison will be undertaken with respect to these results.

The radii $\hat{r}_0$, $\hat{r}_1$, and $\hat{r}_2$ may lack some precision. They were obtained by mapping the solution of equal sphere packing problem (ESPP) to the solution of SPP. ESPP yields, in general, less precise solutions than SPP because of computational errors due to truncation and their faster rate of propagation for values that are less than 1. However, as this chapter makes no attempt to assess these errors, it is assumed that the precision of $\hat{r}_0$ and $\hat{r}_1$ is 10^{-7} (even though it could be suspected that it does not exceed 10^{-6} since $\hat{r}_0 = \hat{r}_1 = 3.0000003$ for $n = 13$). Let $\hat{r} = \min\{\hat{r}_0,\ \hat{r}_1,\ \hat{r}_2\}$. Evidently, $\hat{r}$ has the same precision as the least precise of $\hat{r}_0$, $\hat{r}_1$, and $\hat{r}_2$. On the other hand, r_{H}, $\mathrm{H} \in \mathcal{H}$, are suspected to be precise to 10^{-13}, as can be inferred from r_{H}, $\mathrm{H} \in \mathcal{H}$, for $n = 13$. A valid comparison of r_{H} to $\hat{r}$ requires either truncating or rounding off r_{H} to 10^{-7}. However, regardless of the adopted comparison method, the proposed approach improves $\hat{r}$ for $n = 3-6,\ 9-14, 31, 32, 38$, and 39.

Finally, in many instances, starting LS with a non-feasible solution causes it to stop the search and return an infeasible solution. The number of random initial solutions that needed to be investigated prior to obtaining one that led to a local minima reached 517 in one instance. This number was, however, much less in most instances and averaged 7 for the k_M iterations of M-RLS over and 3 per neighbor for the VNS based heuristics over all runs.

Table 3.2 is a summary of Table 3.1. It reports the number of times r_{H} is less than $r_{\text{H'}}$, (H,H') $\in (\mathcal{H} \cup \{\hat{H}_1, \hat{H}_2, \hat{H}_3\})^2$, with the diagonals giving the number of times H is the only heuristic where in the best upper bound is obtained, and the number in parentheses is the number of times H reaches the best upper bound but is not the only heuristic where it is attained. It highlights the role of the VNS in the search. Most of the best solutions are obtained using VNS based heuristics. In fact, both SLS and M-RLS rarely outperform the VNS heuristics. Table 3.2 further shows that M-VNS-RLS is the best heuristic; that is, its multi-start strategy coupled with its variable neighborhood search allowed it to reach local minima that are closer to the global minimum than the other proposed heuristics.

Table 3.2: Number of Times $r_{\text{H}} < r_{\text{H'}}$

$\downarrow r_{\text{H}} \quad r_{\text{H'}} \longrightarrow$	r_{SLS}	$r_{\text{VNS-SLS}}$	$r_{\text{M-RLS}}$	$r_{\text{VNS-RLS}}$	$r_{\text{M-VNS-RLS}}$	$\hat{r}_0$	$\hat{r}_1$	$\hat{r}_2$
r_{SLS}	0(1)	0	6	1	1	5	4	0
$r_{\text{VNS-SLS}}$	44	1(4)	35	16	3	22	15	10
$r_{\text{M-RLS}}$	42	7	0(0)	4	1	18	12	9
$r_{\text{VNS-RLS}}$	47	12	34	0(4)	0	24	16	12
$r_{\text{M-VNS-RLS}}$	47	28	37	23	0(4)	24	16	13
$\hat{r}_0$	43	26	30	24	24	0(3)	0	21
$\hat{r}_1$	44	33	36	32	32	32	13(5)	29
$\hat{r}_2$	35	25	26	23	22	14	4	3(2)

3.4.2 Feasibility of the initial solution

Table 3.2 shows that the number of times $r_{\text{M-RLS}} < r_{\text{SLS}}$ is 42 whereas the number of times $r_{\text{SLS}} < r_{\text{M-RLS}}$ is only 6 (corresponding to $n = 3$, 23, 25, 39, 41, 43). This discrepancy is expected and should be interpreted with care since $r_{\text{M-RLS}}$ is the best radius over k_M runs of LS; though, it remains true that starting LS from a feasible solution is not always necessary. For example, the cases where $n = 26$ and 27 require that LS be started from a very good quality solution. In fact, starting them from a feasible solution constructed using our naive approach leads to radii that are very far from the best-known upper bounds. Our computational investigation show that starting LS with a lattice configuration yields radii that match the best-known upper bounds for those two cases. However, these results were omitted from Table 3.1 for consistency. On the other hand, the LS failed to converge to a local optimum for those two instances when fed with randomly generated solutions. M-RLS was allowed up to 1000 restarts for each of the k_M iterations. Subsequently, it is recommended that the LS search be started from multiple solutions, with a subset of them being feasible and a subset being randomly generated. This strategy guarantees the convergence of M-SLS to a feasible local minimum while it allows it to investigate the existence of other local minima that might be closer to the global minimum. Therefore, this hybrid strategy is used for larger value of n. We also run our code for $n = 26$ and $n = 27$, using the starting solution obtained by $n = 28$. We get the following results: Tables 3.3, 3.4.

Table 3.3: VNS with initial obtained by hybrid strategy

For	$r_{\text{M-RLS}}$	$r_{\text{VNS-RLS}}$	$r_{\text{M-VNS-RLS}}$
n=26	3.747405776527507	3.747405776527507	3.747405776527507
n=27	3.816151203682777	3.846600560763699	3.816151203682776

Table 3.4: VNS with initial point obtained as for n=28 by removing 1 or 2 centers at random

For	$r_{\text{M-RLS}}$	$r_{\text{VNS-RLS}}$	$r_{\text{M-VNS-RLS}}$
n=26	3.788751680052301	3.747405776527507	3.747405776527507
n=27	3.817290771971196	3.846600560763699	3.846600560763699

3.4.3 Utility of the diversification strategies

The **first** diversification strategy consists in varying Δ. Figures 3.3 to 3.8 illustrate the impact is running when the VNS heuristics with and without fixing a coordinate of one of the spheres. They display $\frac{1}{r_{\mathrm{H}}}$, H $\in$ {VNS-SLS, VNS-RLS, M-VNS-RLS}, as a function of n for different neighborhood sizes (measured in terms of Δ). They confirm the importance of varying Δ in many instances such as n=13, 19-21, 24, 31, 37, 40, 45-49 of Figure 3.3. However, there is no general rule of thumb on how to select the "best" Δ value. Opting for a larger neighborhood is not necessarily better than using a smaller one. Investigating a large neighborhood is equivalent to too much perturbation of the current feasible solution (which itself corresponds to a local minima). In fact, large perturbations induce "large" infeasibility (i.e., overlap). When the VNS heuristics are run without fixing a coordinate of one of the spheres, opting for small Δ values seems more judicious (in the instances where the size of the neighborhood matters). In such cases, using $\Delta = 0.1$ and 0.2 seem reasonable. However, none of these values of consistently dominate the others. This rule holds when the VNS heuristics are run with the coordinate of one of the spheres being fixed in all instances, except for $n = 24$ in Figure 3.6.

The **second** diversification strategy entails in varying T. Starting from the same initial solution, each VNS heuristic H $\in$ {VNS-SLS, VNS-RLS, M-VNS-RLS} was run twice: once with $T = 6$ and once with $T = 12$. Let $r_{\mathrm{H}}^{(6)}$ and $r_{\mathrm{H}}^{(12)}$ denote the respective resulting radii. The results are summarized in Table 3.5; column 1 indicates whether the algorithms are run with or without fixing a coordinate of one of the spheres. Columns 2 and 3 specify the neighborhood size Δ and the heuristic H. Columns 4-6 display η_1, η_2, η_3, the number of times $r_{\mathrm{H}}^{(12)} > r_{\mathrm{H}}^{(6)}$, $r_{\mathrm{H}}^{(12)} = r_{\mathrm{H}}^{(6)}$, and $r_{\mathrm{H}}^{(12)} < r_{\mathrm{H}}^{(6)}$, respectively. Column 7 reports η_4 the number of times H fails to

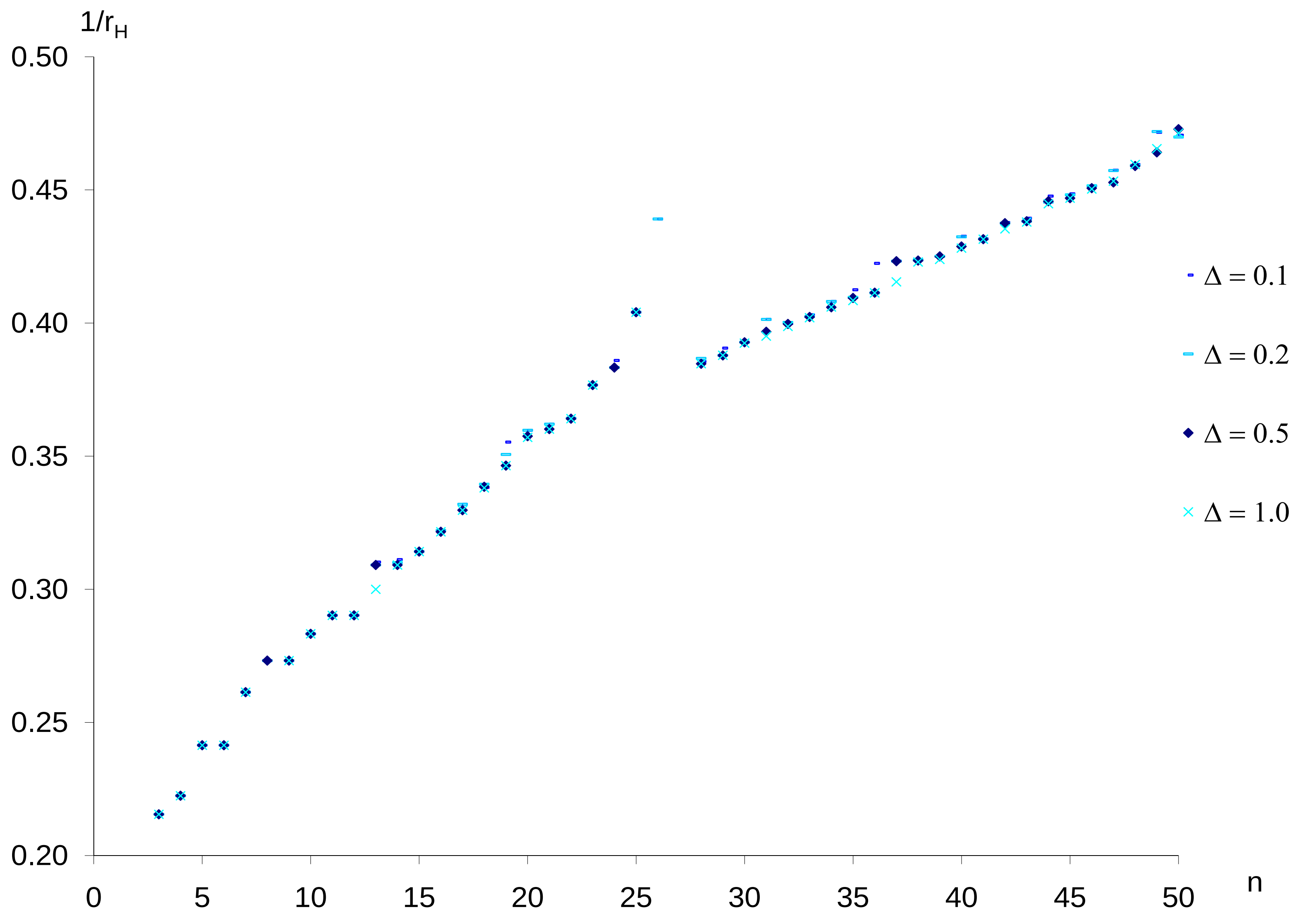

Figure 3.3: Impact of the neighborhood size on VNS-SLS^{N}.

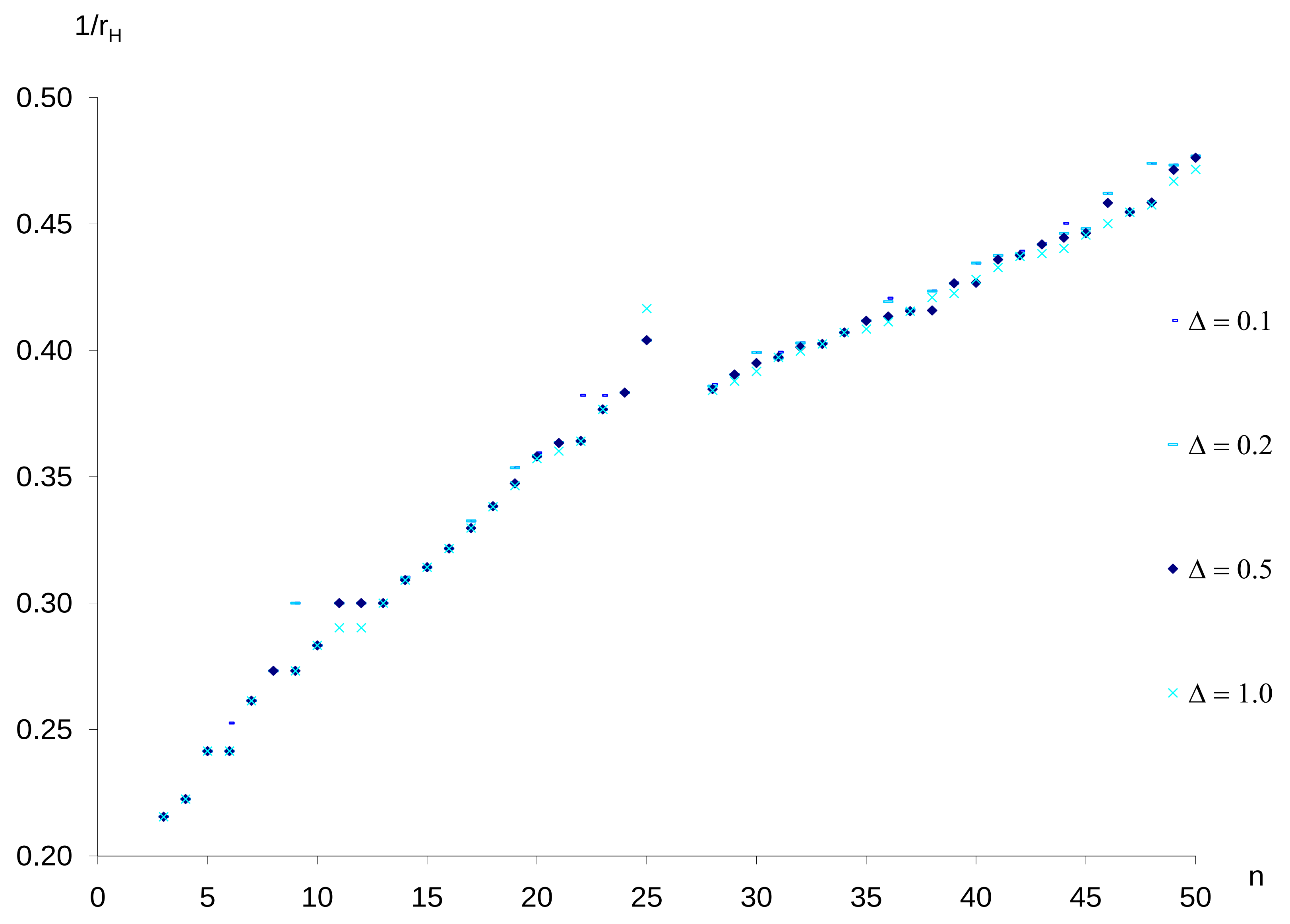

Figure 3.4: Impact of the neighborhood size on VNS-RLS^N.

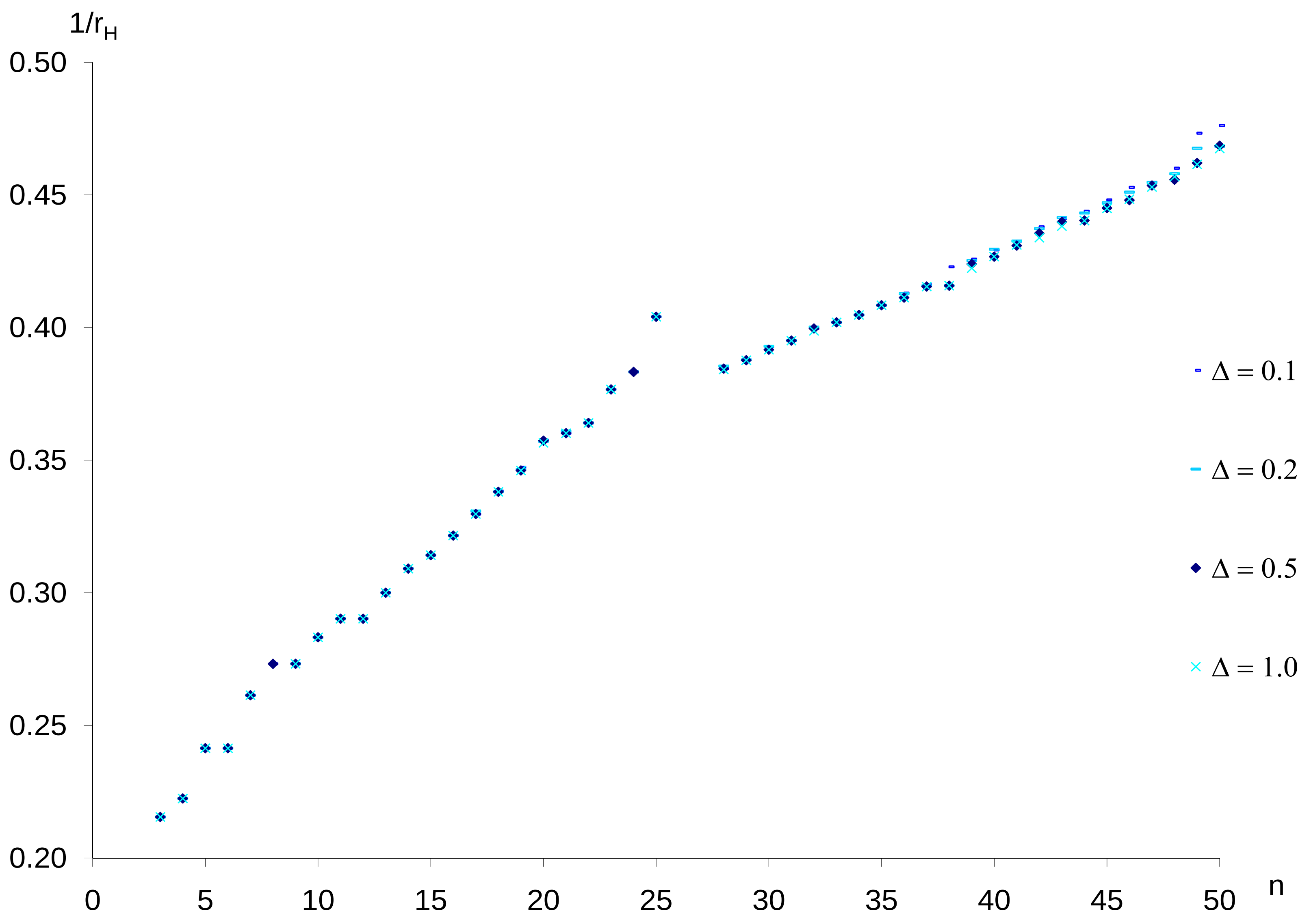

Figure 3.5: Impact of the neighborhood size on M-VNS-RLS$^{\text{N}}$.

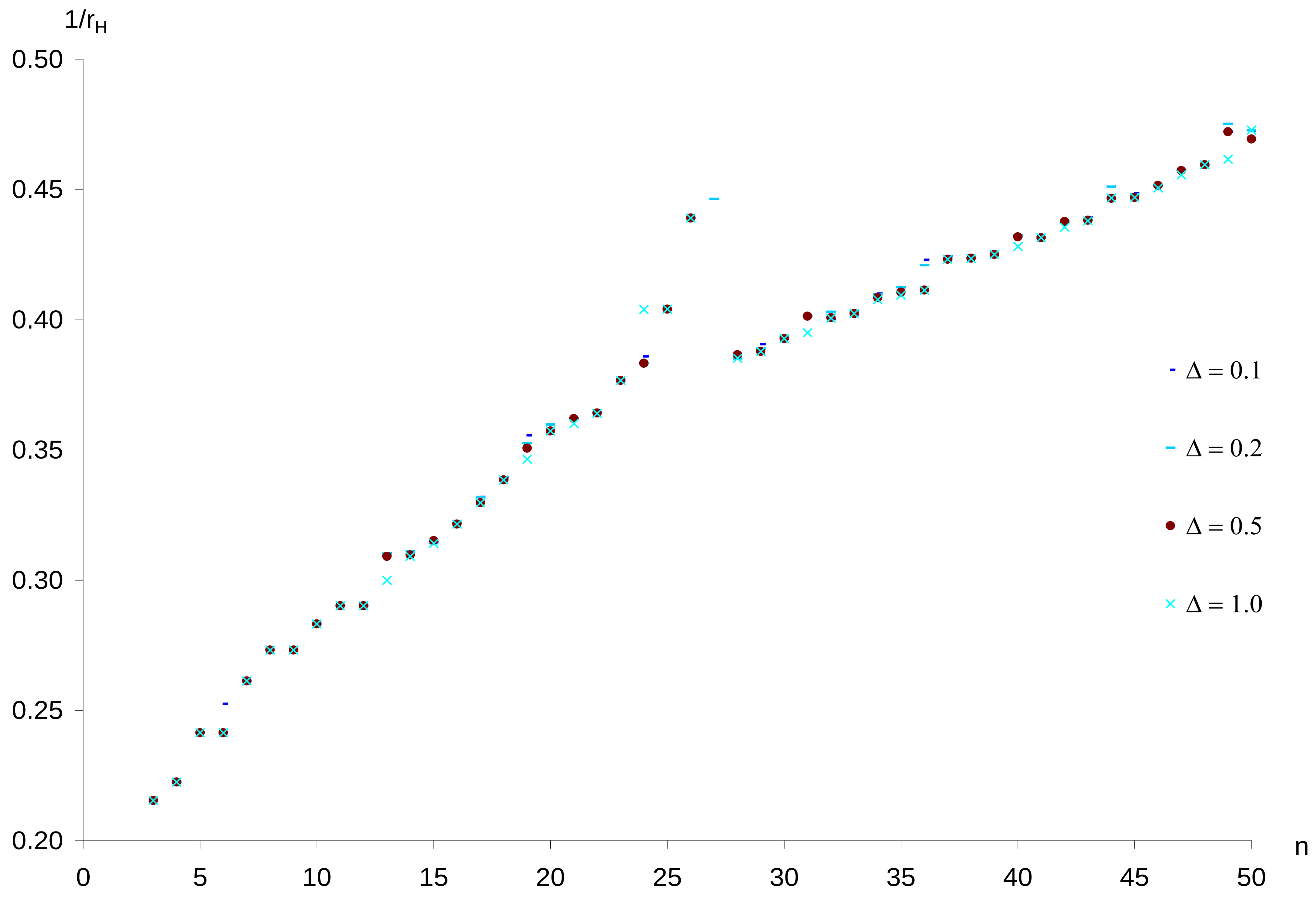

Figure 3.6: Impact of the neighborhood size on VNS-SLS^F.

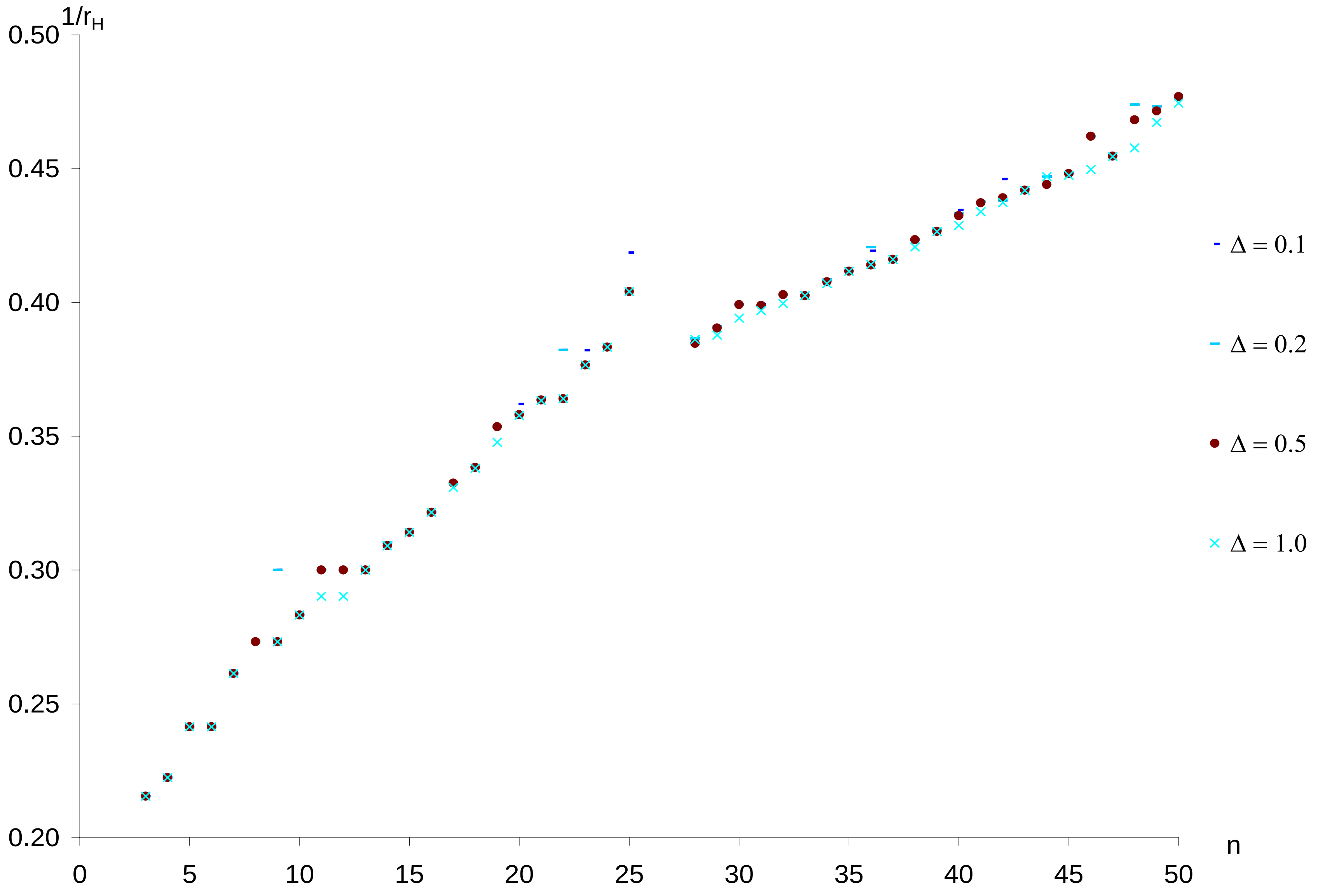

Figure 3.7: Impact of the neighborhood size on VNS-RLS^{F}.

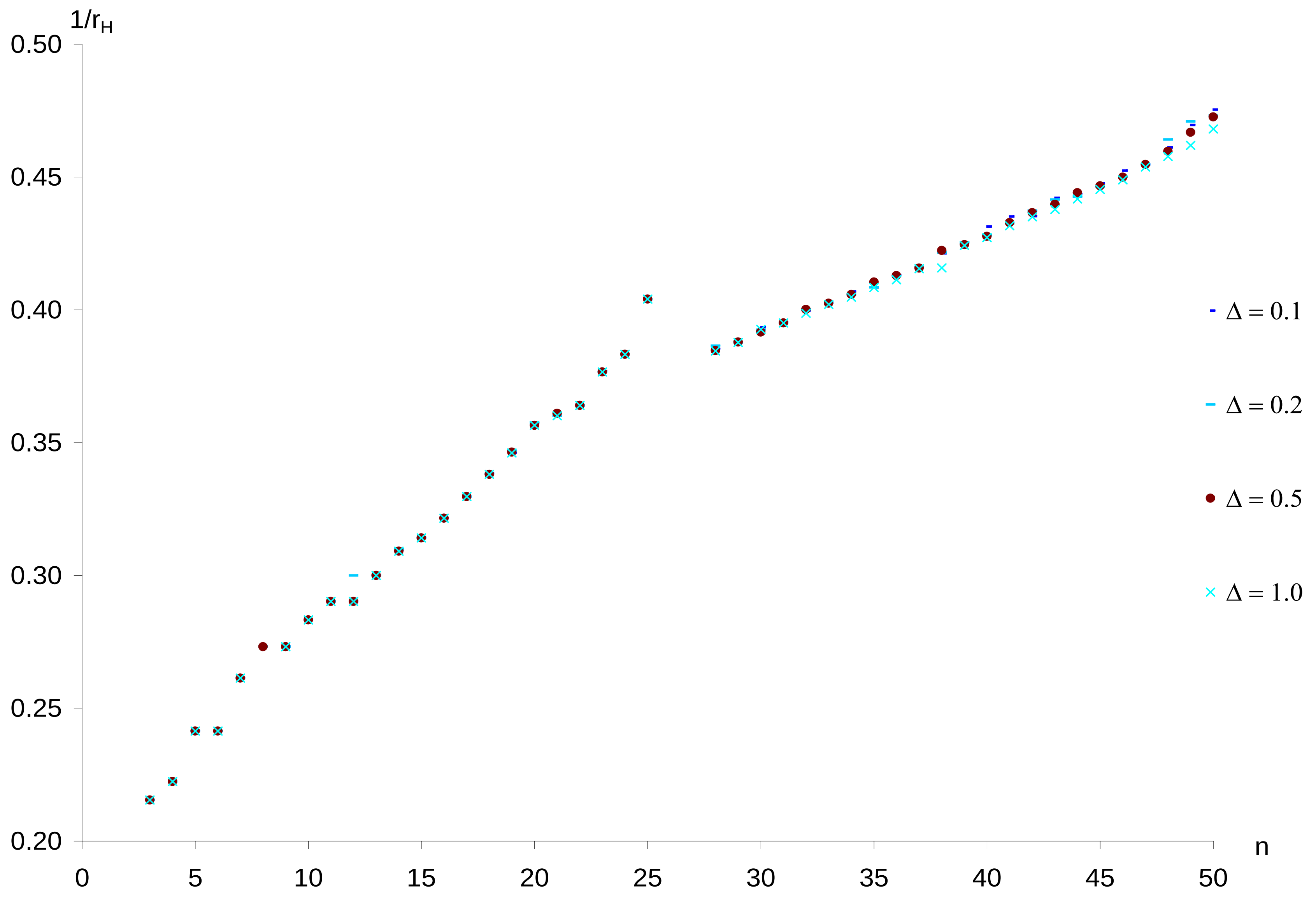

Figure 3.8: Impact of the neighborhood size on M-VNS-RLS^{F}.

identify a feasible solution when $T = 6$. Finally, Columns 8 and 9 report the average and maximum deviations of $r_{\mathrm{H}}^{(6)}$ from $r_{\mathrm{H}}^{(12)}$ over the cases with $r_{\mathrm{H}}^{(6)} < r_{\mathrm{H}}^{(12)}$ whereas the last two columns report the average and maximum deviations of $r_{\mathrm{H}}^{(12)}$ from $r_{\mathrm{H}}^{(6)}$ over the cases with $r_{\mathrm{H}}^{(12)} < r_{\mathrm{H}}^{(6)}$.

Results in Table 3.5 suggest that varying Δ can be successfully used as a diversification strategy of VNS. Using $T = 6$ yields more frequent better-quality solutions than using $T = 12$. However, the average and maximum improvements brought up by using $T = 12$ when $r_{\mathrm{H}}^{(12)} < r_{\mathrm{H}}^{(6)}$ are more sizeable than their counterparts when $r_{\mathrm{H}}^{(12)} = r_{\mathrm{H}}^{(6)}$.

The cases when the magnitude of T does not affect the solution quality are numerous, and the number is more important when H is run with a fixed coordinate than when it is not fixed and for $\Delta = 1.0$ than for $\Delta = 0.5$. It also depends on the heuristic H (most probably on the quality of the initial solution fed to H).

When $\Delta = 1.0$, the VNS heuristics fail in 35% of the instances to identify a feasible solution when $T = 6$, for either the F or N versions of the algorithm. This percentage is twice as large (16%) for the F version than for the N version (8%) when $\Delta = 0.5$. This suggests that using $\Delta = 1.0$ while fixing a coordinate of one of the spheres may result in infeasibility that cannot be resolved by the VNS within the preset time limit $T = 6$; hence, none of the solutions fed to LS would yield a feasible solution.

The **third** diversification strategy consists in running each of the VNS heuristics twice: starting from the same initial solution, H $\in$ {VNS-SLS, VNS-RLS, M-VNS-RLS}, and run once with and once without fixing a coordinate of one of the spheres; denoted by H^{F} and H^{N} respectively. The results are summarized by Table 3.6 . Columns 1 and 2 specify the neighborhood size Δ and the heuristic H.

Table 3.5: Impact of T

							$r_{\mathrm{H}}^{(12)} - r_{\mathrm{H}}^{(6)}$		$r_{\mathrm{H}}^{(6)} - r_{\mathrm{H}}^{(12)}$	
	Δ	H	η_1	η_2	η_3	η_4	Average	Maximum	Average	Maximum
F	0.5	VNS-SLS	23	14	7	4	0.0006797964315658	0.0047150070802350	0.0000057821374355	0.0001696240956520
		VNS-RLS	23	14	7	4	0.0010415722555616	0.0075134699484810	0.0000000000000015	0.0000000000000300
		M-VNS-RLS	34	9	3	4	0.0008022943553068	0.0065668290765730	0.0000128310629656	0.0005645667703930
F	1.0	VNS-SLS	11	12	6	17	0.0009741013792953	0.0207184337198160	0.0000000000000051	0.0000000000000590
		VNS-RLS	14	9	6	17	0.0006410612097868	0.0032101243210140	0.0000000000000094	0.0000000000001630
		M-VNS-RLS	6	17	6	17	0.0000494666703618	0.0010025105653920	0.0000000000000058	0.0000000000001470
N	0.5	VNS-SLS	14	17	5	7	0.0001868529703426	0.0027082005313520	0.0000249052585654	0.0008965893082480
		VNS-RLS	16	15	5	7	0.0005693284342984	0.0039924776935160	0.0007489865918569	0.0267949192430750
		M-VNS-RLS	14	17	5	7	0.00020905116113104	0.002094714645463	0.0000088700368954	0.0003193213279810
N	1.0	VNS-SLS	8	12	7	16	0.0001266946270644	0.0012782688071870	0.0000502741971153	0.0005316148398080
		VNS-RLS	4	22	1	16	0.0002157503380220	0.0022734007341170	0.0000000000000004	0.0000000000000100
		M-VNS-RLS	5	19	3	16	0.0000000000000043	0.0000000000000820	0.0000000000000043	0.0000000000001070

Columns 3 to 5 display $\eta_1,\ \eta_2,\ \eta_3$, the number of times $r_{\mathrm{H^N}} < r_{\mathrm{H^F}},\ r_{\mathrm{H^F}} = r_{\mathrm{H^N}}$, and $r_{\mathrm{H^N}} > r_{\mathrm{H^F}}$, respectively. Column 6 reports η_4, the number of times the VNS heuristic fails to identify a feasible solution. Finally, Columns 8 and 9 report the average and maximum deviations $r_{\mathrm{H^N}} - r_{\mathrm{H^F}}$ when $r_{\mathrm{H^F}} < r_{\mathrm{H^N}}$, whereas Columns 10 and 11 give the average and maximum deviations $r_{\mathrm{H^F}} - r_{\mathrm{H^N}}$ when $r_{\mathrm{H^F}} > r_{\mathrm{H^N}}$.

These results show that fixing the coordinate of one of the spheres is neither always preferable nor undesirable. In fact, the maximum deviations of $|r_{\mathrm{H^F}} - r_{\mathrm{H^N}}|$ can reach 0.012: a very important variation for the problem at hand. Subsequently, in many instances, reducing the search space (with the objective of eliminating symmetry and related issues) is not advisable. Thus, a successful search strategy should opt for testing both approaches of the problem, as is further elucidated by the detailed results for $\mathrm{H^F}$ and $\mathrm{H^N}$ in Table 3.7.

3.4.4 Utility of the VNS and the LS

The best radii obtained by $\mathrm{H^F}$ and $\mathrm{H^N}$ when $\Delta = 0.5$ and $T = 12$ is given in Table 3.7. Column 1 indicates the problem size n. Columns 2 and 3 display r_{SLS} and $r_{\mathrm{M\text{-}RLS}}$. Columns 3 to 5 report $r_{\mathrm{H}}^{\mathrm{F}}$, the best radius r obtained by heuristic $\mathrm{H^F}$, H $\in$ {VNS-SLS, VNS-RLS, M-VNS-RLS}, whereas Columns 6 to 8 display the same information but for $\mathrm{H^N}$.

The result in Table 3.7 confirm that initiating LS from a feasible solution always yields a local minimum. This is obviously not the case for infeasible solutions. For instance, in the 48 instances tested above, LS failed to reach a local minimum when started from a large set of randomly generated solutions. It is also showed that the LS obtains a better solution when started with a feasible solution in ten out of 48 instances, with the maximum deviation of r_{SLS} from $r_{\mathrm{M\text{-}RLS}}$ reaching

Table 3.6: Impact of Fixing a Coordinate of One of the Spheres

						$r_{H^N} - r_{H^F}$		$r_{H^F} - r_{H^N}$	
Δ	H	η_1	η_2	η_3	η_4	Average	Maximum	Average	Maximum
1.0	VNS-SLS	19	15	10	4	0.0004974439053977	0.0077303518392241	0.0000902961122576	0.0038619636613130
	VNS-RLS	22	11	11	4	0.0008949954878760	0.0066950721342630	0.0003042420018562	0.0124798832617420
	M-RLS-VNS	17	21	6	4	0.0002361300859217	0.0018611165082080	0.0000094867061752	0.0004174150713850
0.5	VNS-SLS	23	18	5	2	0.0008214480754788	0.0080898502012480	0.0000780921804996	0.0034180268264650
	VNS-RLS	22	16	8	2	0.0011552657855988	0.0097252610557670	0.0000134694644134	0.0004769662880320
	M-RLS-VNS	30	11	5	2	0.0008658800087198	0.0065668290765730	0.0000190941435287	0.0008172609000440

Table 3.7: Effect of Fixing the Position of a Sphere on the Quality of the Solution with $\Delta = 0.5$ and $T = 12$

			With fixing			Without fixing		
n	r_{SLS}	$r_{\text{M-RLS}}$	$r_{\text{VNS-SLS}}$	$r_{\text{VNS-RLS}}$	$r_{\text{M-VNS-RLS}}$	$r_{\text{VNS-SLS}}$	$r_{\text{VNS-RLS}}$	$r_{\text{M-VNS-RLS}}$
3	2.15470053824456	2.15470053837475	2.15470053824456	2.15470053837425	2.15470053837425	2.15470053824456	2.15470053837425	2.15470053837425
4	2.41421356237309	2.22474487139144	2.22474487138673	2.22474487138675	2.22474487138675	2.22474487138693	2.22474487138683	2.22474487138681
5	2.41421356237298	2.41421356237147	2.41421356236963	2.41421356236839	2.41421356236831	2.41421356236870	2.41421356236878	2.41421356236823
6	2.52752523165194	2.41421356237309	2.41421356237253	2.41421356237243	2.41421356237094	2.41421356237227	2.41421356237169	2.41421356237055
7	2.61368668160301	2.61524242091076	2.61368668160272	2.61368668160289	2.61368668160243	2.61368668160219	2.61368668160295	2.61368668160281
8	2.73205080757142	2.73205080756888	2.73205080756887	2.73205080756887	2.73205080756887	2.73205080756887	2.73205080756887	2.73205080756887
9	2.74182565152031	2.73205080756887	2.73205080756858	2.73205080756884	2.73205080756793	2.73205080756874	2.73205080756887	2.73205080756668
10	2.83246456105392	2.83246456105556	2.83246456105391	2.83246456105391	2.83246456105391	2.83246456105391	2.83246456105391	2.83246456105391
11	2.90211303259061	2.90211303259030	2.90211303259030	2.99999999999989	2.90211303259030	2.90211303259030	2.99999999999982	2.90211303259030
12	2.99941691897225	2.90211303259030	2.90211303259030	2.99999999999975	2.90211303259030	2.90211303259030	2.99999999999981	2.90211303259030
13	3.10221395734388	3.00000000000000	3.09114544488871	2.99999999999991	2.99999999999972	3.09114544488871	2.99999999999981	2.99999999999950
14	3.11144281851829	3.09736179246790	3.09736179246790	3.09114544488871	3.09114544488871	3.09114544488871	3.09114544488871	3.09114544488867
15	3.32152692721521	3.14164262494871	3.15220026555598	3.14164262494871	3.14164262494871	3.14164262494870	3.14164262494871	3.14164262494871
16	3.22035762742487	3.21588315803967	3.21588315803967	3.21588315803967	3.21588315803967	3.21588315803967	3.21588315803967	3.21568303201004
17	3.31921100101264	3.29684043850944	3.29715029402540	3.32517878952614	3.296840438509440	3.29715029402540	3.29680836874176	3.29680836874176
18	3.43690158448033	3.38251273518736	3.38473980290167	3.38352323931557	3.38084090740455	3.38473980290167	3.38352323931557	3.38084090740455
19	3.55861664021641	3.47786806710072	3.50649304551939	3.53539962883257	3.46435524031985	3.46435524031985	3.47364779702559	3.46116202704004
20	3.59742989177187	3.57665233706004	3.57317588015077	3.58020224117702	3.56548622298350	3.57487212110729	3.58020224117701	3.57365883198394
21	3.62033409540338	3.61070943656382	3.62030787516073	3.63568047804109	3.61070943656382	3.60153022804217	3.63363147125231	3.60153022804217
22	3.68953464150661	3.64051666844219	3.64192954910622	3.64051666844219	3.64051666844219	3.64051666844219	3.64192955447838	3.64051666844219
23	3.76703739927985	3.76879846847560	3.76703739927985	3.76654989681805	3.76654989681805	3.76703739927985	3.76654989681805	3.76654989681805
24	4.22574928847579	3.83289514992629	3.83289390622137	3.83289390622137	3.83281214860800	3.83289390622137	3.83289390622137	3.83281214860673
25	4.04050812837879	4.18881752094069	4.04050812837879	4.04050812837879	4.04050812837879	4.04050812837879	4.04050812837879	4.04050812837879
26	4.39215002079962		4.39080038861019					
27	4.46410161513775							
28	4.05290053391191	3.85938265121346	3.86582129025475	3.84677122436886	3.84677122436886	3.84677122436886	3.84677122436886	3.84509642706231
29	3.90588517026782	3.88966248881778	3.87839027754598	3.90457084552566	3.87843617135428	3.87843617135428	3.90457084552566	3.87708910315835
30	3.93900318427918	3.92445917487543	3.92746450746635	3.99162636103909	3.91649166155428	3.92746450746635	3.94982913114897	3.91649166155428
31	4.01340436656033	3.97113053380349	4.01340436655556	3.98850892264477	3.95088896242510	3.96727513653610	3.97331290576204	3.95099049486861
32	4.27254890158653	4.00894046081310	4.00764532440897	4.02906991302087	4.00188284558907	3.99661151602936	4.01262649482391	3.99650463579512
33	4.08580036058113	4.02697939807711	4.02405977925337	4.02513119188639	4.02513119188639	4.02288135207819	4.02513119188639	4.01990091595852
34	4.10040899215648	4.04944723850525	4.08482441338340	4.07683683181969	4.04944723850525	4.05908254156022	4.07045397857175	4.04771997123058
35	4.13002114560296	4.08444990930670	4.10582201819603	4.11682595453275	4.08444990930670	4.09432596421864	4.11682595453275	4.08440574075337
36	4.30716585797470	4.14241224205426	4.11383912694139	4.13981328995080	4.12914477837374	4.11383912694139	4.13385293937593	4.11298932968653
37	4.24424648243793	4.16096800652210	4.23241893670340	4.16096800652210	4.15696571952974	4.23208477038344	4.15478125199120	4.15478125199120
38	4.23715102793316	4.23461234366052	4.23597802016207	4.23461234366052	4.22333755081395	4.23448099686564	4.15773497480818	4.15766926004822
39	4.25333789139973	4.27151398035863	4.25034151073013	4.26513995084063	4.24536170211884	4.25034151073013	4.26513995084063	4.23938145003991
40	4.38558965541323	4.34224220302311	4.31860576096985	4.32455096836581	4.27706783885791	4.28741505000694	4.26746039311058	4.26746039311058
41	4.31673932506242	4.37544223244230	4.31499920733667	4.37213616597835	4.32849297421721	4.31499920733667	4.35904934368174	4.30884069402462
42	4.37739382983185	4.38859740716231	4.37738425016750	4.39107436026373	4.36554388773070	4.37485649842280	4.37541322421017	4.35524264727754
43	4.39892791988379	4.42059573559044	4.38208253229267	4.41932860365510	4.39755136379951	4.38208253229267	4.41934200836812	4.39806052837831
44	4.63248403341084	4.43832644664312	4.46592125867765	4.44088832161426	4.43832644664312	4.45745720730599	4.44565798449458	4.40388868867003
45	4.49709897273965	4.48132644812954	4.46921272862671	4.48128019704679	4.46641782049140	4.46921272862671	4.46350443991294	4.45075663296053
46	4.54196325546583	4.62091267720554	4.51531840602923	4.62091267720554	4.49999576911328	4.50636205851480	4.58303242931251	4.48035842049339
47	4.59256828247835	4.54766625099119	4.57260789490155	4.54703260392380	4.54703260392380	4.52777880318190	4.54703260392380	4.53556984245498
48	4.74031835202101	4.65073005954271	4.59521585427709	4.68197552549096	4.59623053636001	4.58977916293008	4.58472291493329	4.55711696552466
49	4.83383651985818	4.70368559799948	4.72170144421721	4.71557841838051	4.66875731783470	4.64080294220473	4.71386677809460	4.62051643160067
50	4.76300934968712	4.74017473378151	4.69376374175179	4.76893782316805	4.72622078855716	4.72794401001644	4.76209491853735	4.68541082329544
Number of times $r_{\mathrm{H}}^{F} < r_{\mathrm{H}}^{\mathrm{N}}$			5	8	5			
Number of times $r_{\mathrm{H}}^{\mathrm{N}} < r_{\mathrm{H}}^{F}$						23	22	30

0.148309392561900. On the other hand, LS obtains a better solution when started from randomly generated solutions in 36 out of 48 instances with the maximum deviation being 0.392854138549501. It seems therefore judicious to include an initial feasible solution among the solutions investigated by LS to guarantee its convergence to a local optimum and its improved overall performance.

3.4.5 Comparison of the diversification strategies

VNS-SLS versus SLS

As expected, $r_{\text{SLS}} \geq r_{\text{VNS-SLS}}$. Applying VNS$^{\text{F}}$ to the local optimum obtained by SLS improved r_{SLS} in 44 out of 48 instances. The improvement reaches 0.392855382254420, and averages 0.055338061287952 over the 48 instances. Similarly, VNS$^{\text{N}}$ improves r_{SLS} in 43 out of 48 instances. The improvement reaches 0.392855382254420, and averages 0.063762149389844 over the 48 instances.

VNS-RLS versus VNS-SLS

VNS$^{\text{F}}$-RLS yields better solutions than VNS$^{\text{F}}$-SLS in 25 out of 48 instances, with the maximum deviation of $r_{\text{VNS-SLS}}$ from $r_{\text{VNS-RLS}}$ reaching -0.105594271176310. The opposite case occurs in 16 instances with the maximum deviation equaling 0.091145444888800. The overall average deviation is -0.013753483385387. Similarly, VNS$^{\text{N}}$-RLS yields better solutions than VNS-SLS in 27 out of 48 instances, with the maximum deviation of $r_{\text{VNS-SLS}}$ from $r_{\text{VNS-RLS}}$ reaching -0.097886967409520. The opposite case occurs in 12 instances with the maximum deviation equaling 0.091145444888900. The overall average deviation is -0.010181470957522. This suggests that VNS should be started from a feasible solution when not used in conjunction with a multi-start strategy.

VNS-RLS versus M-RLS

The hybridization of VNS and LS can provide a powerful search method if it addresses the two competing goals of meta-heuristics: exploration and exploitation. Exploration allows an extensive search of the solution space in order to determine the part of the space that has a higher chance of containing the global optimum whereas exploitation refines the search and focuses on the part of the space that has a high potential of containing the global optimum. Exploration can be herein obtained via multi-start or via VNS whereas exploitation is obtained via LS. Both the multi-start and VNS strive for global optimization while LS strives for local optimization in the global optimum neighborhood. The effectiveness of using VNS and multi-start as the diversification strategies is discussed here.

The findings presented in Table 3.7 does not provide a clear answer to this dilemma. It suggests using a mixture of the two search strategies. In fact, $r^{\mathrm{F}}_{\text{VNS-RLS}} < r^{\mathrm{F}}_{\text{M-RLS}}$ in 22 out 48 instances, while $r^{\mathrm{F}}_{\text{VNS-RLS}} > r^{\mathrm{F}}_{\text{M-RLS}}$ in 18 out 48 instances. The maximum deviation in the first case equals -0.148309392561900 while it equals 0.097886967409590 in the second case. Its average is 0.004562699793446. Similarly, $r^{\mathrm{N}}_{\text{VNS-RLS}} < r^{\mathrm{N}}_{\text{M-RLS}}$ in 28 out 48 instances while $r^{\mathrm{N}}_{\text{VNS-RLS}} > r^{\mathrm{N}}_{\text{M-RLS}}$ in 15 out 48 instances. The maximum deviation in the first case equals -0.148309392561900 while it equals 0.097886967409520 in the second case. Its average is -0.006120725568965. These different magnitudes of improvement over all instances of both versions of the algorithms suggest that both diversification and intensification of the search are important; one dose not predominate the other in any case.

M-VNS-RLS versus VNS-RLS

Comparison of Columns 5 and 6 and Columns 8 and 9 of Table 3.7 highlights the importance of the diversification of the search by starting VNS-RLS from a number of randomly generated solutions. In fact, the additional investigation brings forth sizeable improvements. For VNS$^{\mathrm{F}}$, an improvement is registered in 32 out of 48 instances with the maximum improvement reaching 0.120916908092260 and averaging 0.021911628194197 over the 48 instances. The same trend is observed for VNS$^{\mathrm{N}}$ where an improvement is registered in 37 out of 48 instances with the maximum improvement reaching 0.102674008819120 and averaging 0.019515894278337 over the 48 instances.

M-VNS-RLS versus M-RLS

Comparison of Column 3 to Columns 6 and 9 of Table 3.7 quantifies the importance of diversifying the search by applying VNS to each of the k_M local optima obtained by M-RLS. The larger the discrepancy between $r_{\text{M-VNS-RLS}}$ and $r_{\text{M-RLS}}$, the more important the role of VNS is, in identifying the global optima. For VNS$^{\mathrm{F}}$, an improvement is observed in 36 out of 48 instances with the maximum improvement reaching 0.148309392561900 and averaging 0.014861185920063 over the 48 instances. Similarly, for VNS$^{\mathrm{N}}$, an improvement is observed in 42 out of 48 instances with the maximum improvement reaching 0.148309392561900 and averaging 0.022481100856624 over the 48 instances.

3.5 Conclusion

A variable neighborhood search approach is proposes in this chapter, to solve the problem of packing n unitary spheres into the smallest containing sphere S where the objective is to identify the radius of S and a feasible configuration of the unitary spheres within S. The approach follows the recent research trends of combining search heuristics with non-linear programming tools and of balancing exploitation and exploration. Indeed, it ensures exploitation by applying a local search based on a sequential quadratic programming algorithm with a non-monotone line search (as a non-linear programming solver) and exploration by applying a variable neighborhood search and a multi-start strategy. The approach provides accurate results with a precision of 10^{-13}. It can be extended to the case of non-identical spheres and to the packing of spheres into other 3-dimensional shapes.

Chapter 4

Packing Spheres in a Cube

4.1 Introduction

In this chapter, the adaptation of the VNS to the problem of packing sphere in a cube is explored. An introduction to the VNS has been given in section 2.1 of chapter 2, so we will not repeat its rules here.

Sphere packing in a cube is an optimization problem, that could be classified either as continuous and or as discrete. The positions of the spheres are presented in continuous variables, whereas the structure of an optimal configuration is discrete [49]. A successful solution technique should tackle these two aspects simultaneously.

Accordingly, we approximately solve the problem of packing unit spheres into the smallest cube (PSC) using a variable neighborhood search (VNS) that combines these two aspects. A VNS addresses the discrete aspect of the PSC by shaking one or more spheres and its continuous aspect by applying a non-linear programming (NLP) optimizer that identifies a local optimum within the neighborhood of the fed solution. In addition, it yields local optima that are nearer to the global optimum, in a more straightforward manner than do other meta-heuristics. The VNS has successfully been applied to solve many problems, including integer and continuous optimization problems [52].

For 3D packing of spheres the literature is not as extensive as for solving correspond 2D packing. However, it is clear that mathematical programming formulations could easily be extended from 2D to 3D. As far as we know, there is only one journal paper considering 3D packing of spheres into a cube by Birgin and Sobral [13]. The authors considered the objects in different shapes as triangle, circle, square and strips. More detailed description of their work is given in chapter 3. Other two references that we will use in computational results to compare with, we take from websites [4, 5] that just report the best known values. However, it is not clear what methods they used to report those values.

4.2 Mathematical model

PSC comprises of packing n identical spheres, of radius one, without overlap into the smallest containing cube C. The goal is to search for the best packing of the n spheres into C, where the best packing minimizes w, the length of the side of C. According to the typology of Wascher et al. [79], PSC is a three-dimensional variant of the Open Dimension Problem, since all small items (which are spheres) have to be packed and the extension of the large object (which is a cube) is not given but has to be minimized. PSC is equivalent to finding the coordinates (x_i, y_i, z_i) of every sphere $i,\ i = 1, \dots, n$, and the side w of C. There are several equivalent nonlinear programming formulations of PSC. The simplest one is given in Costa et al. [7] where objective is to maximize the small radius of sphere.

$$\min \quad w \tag{4.1}$$

$$\text{s.t.} \tag{4.2}$$

$$(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 \geq w \qquad 1 \leq i < j \leq n, \tag{4.3}$$

$$0 \leq x_i \leq 1 \qquad i = 1, \ldots, n, \tag{4.4}$$

$$0 \leq y_i \leq 1 \qquad i = 1, \ldots, n, \tag{4.5}$$

$$0 \leq z_i \leq 1 \qquad i = 1, \ldots, n, . \tag{4.6}$$

In this chapter we will use the following model [13]

$$\min \quad w \tag{4.7}$$

$$\text{s.t.} \tag{4.8}$$

$$(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 \geq 4 \qquad 1 \leq i < j \leq n, \tag{4.9}$$

$$1 \leq x_i \leq w - 1 \qquad i = 1, \ldots, n, \tag{4.10}$$

$$1 \leq y_i \leq w - 1 \qquad i = 1, \ldots, n, \tag{4.11}$$

$$1 \leq z_i \leq w - 1 \qquad i = 1, \ldots, n, \tag{4.12}$$

$$w \geq \underline{w}, \tag{4.13}$$

where $\underline{w} = \sqrt[3]{\frac{4}{3}\pi n}$ is a lower bound on w [13]. The first set of constraints ensures there is no overlap of any pair of distinct spheres. Note that this set of constraints could be replaced with a single one as [13]:

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \max\{0, 4-(x_i-x_j)^2-(y_i-y_j)^2-(z_i-z_j)^2\} = 0, \tag{4.14}$$

The next three sets (4.10 - 4.12) ensure the containment of every sphere within C.

PSC assumes that the bottom leftmost lowest of C coincides with the origin $(0;0;0)$. PSC has an in infinite number of alternative optima caused by symmetrical configurations. Fixing the position of one of the spheres would reduce the search space by eliminating a large number of equivalent solutions [8, 16, 17, 45, 61], without necessarily loosing subspaces including good quality local optima. Herein, we set two of the coordinates of a randomly chosen sphere to 1 so that the solver maintains this sphere adjacent to two edges of the cube during its search. Additional bounding constraints that lexicographically ordered the spheres were investigated but not adopted in this model. Solving PSC via an NLP solver only is generally not successful [5]. Subsequently, PSC is solved by VNS.

4.3 Variable neighborhood search-based algorithm for the PSC problem

The proposed VNS is to solve the PSC problem detailed in Algorithm 2. It is in fact very similar to one described in chapter 3 for solving PSS. It starts its search from a *feasible* solution $\mathbf{u} = (x_1, y_1, z_1, \ldots, x_n, y_n, z_n)$ for PSC with a side length $w_{\mathbf{u}}$. The coordinates of $\mathbf{u}$ by randomly generating by using uniform distribution from the interval $(1, 2\sqrt[3]{n})$. Any overlapping spheres are moved to out-skirt of C. VNS sets the best solution $\mathbf{u}^* = \mathbf{u}$, and its upper bound $w^*_{\mathbf{u}} = w_{\mathbf{u}}$. It initializes its cumulated runtime t is 0 and $k = 1$.

Algorithm 2 Detailed Algorithm of the VNS for PSC

```
Input
  1  A feasible solution $\mathbf{u} = (x_i, y_i, z_i)_{i=1,\dots,n}$, and $w_{\mathbf{u}}$ its side length of $C$.
  2  $T$, the maximal runtime for VNS.
  3  $\bar{k}$, the maximal number of neighborhoods.
Output
  1  A (near-) optimal solution $\mathbf{u}^* = (x_i^*, y_i^*, z_i^*)_{i=1,\dots,n}$, and its side length $w_{\mathbf{u}^*}$.
Algorithm
  1  Set $\mathbf{u}^* = \mathbf{u}$ and $w_{\mathbf{u}^*} = w_{\mathbf{u}}$.
  2  Set the algorithm's cumulated runtime $t$ to 0.
  3  Do while $t \leq T$
    3.1  Set the neighborhood type $k = 1$.
    3.2  Do while $k \leq \bar{k}$
      3.2.1  Generate a random solution $\mathbf{u}'$ from the $k^{\text{th}}$ neighborhood $N_k(\mathbf{u})$ of $\mathbf{u}$.
      3.2.2  Starting from $\mathbf{u}'$, find a local minimum $\mathbf{u}"$ and its side length $w_{\mathbf{u}"}$.
      3.2.3  If $w_{\mathbf{u}"} < w_{\mathbf{u}}$,
                set $k = 1$, $\mathbf{u} = \mathbf{u}"$, and $w_{\mathbf{u}} = w_{\mathbf{u}"}$;
                if $w_{\mathbf{u}"} < w_{\mathbf{u}^*}$, set $\mathbf{u}^* = \mathbf{u}"$, and $w_{\mathbf{u}^*} = w_{\mathbf{u}"}$.
             Else
                set $k = k + 1$.
      3.2.4  Update $t$.
```

The VNS is an iterative procedure with two loops. The outer loop controls the runtime whereas the inner one undertakes the search. At each iteration of the inner loop, the VNS generates a random solution $\mathbf{u}'$ from a neighborhood $N_k(\mathbf{u})$

of $\mathbf{u}$. It finds a local minimum $\mathbf{u}" \in N_k(\mathbf{u})$ by applying NLP to the PSC starting from $\mathbf{u}'$. The selected NLP solver is NLPQLP, a sequential quadratic programming algorithm with a non-monotone line search [63]. The NLPQLP is designed for smooth NLP problems. When the solver aborts because of computational errors caused by inaccurate function or gradient evaluations, a non-monotone line search is activated. Internal restarts are performed in case of errors when computing the search direction due to inaccurate derivatives. Additional automated initial and periodic scaling with restarts are implemented [63].

If $w_{\mathbf{u}"} < w_{\mathbf{u}}$, the current solution $\mathbf{u}$ is updated; i.e., the focus of the search is re-centered on $\mathbf{u}"$, and k is reset to 1. Furthermore, if $w_{\mathbf{u}"} < w_{\mathbf{u}^*}$, then $w_{\mathbf{u}^*}$ is reset to $w_{\mathbf{u}"}$, and $\mathbf{u}^*$ to $\mathbf{u}"$.

On the other hand, if $w_{\mathbf{u}"} \geq w_{\mathbf{u}}$, the neighborhood of $\mathbf{u}$; i.e., k is incremented. A random solution $\mathbf{u}' \in N_k(\mathbf{u})$ is chosen, and the local optimum $\mathbf{u}"$ within $N_k(\mathbf{u})$ is identified. The neighborhood is enlarged until one of the two stopping criteria is met:

∗ A solution $\mathbf{u}" \in N_k(\mathbf{u})$ such that $w_{\mathbf{u}"} < w_{\mathbf{u}}$ is obtained; in which case, k is reset to 1 and the inner loop is repeated.

∗ The number of investigated neighborhoods k reaches the threshold $\bar{k}$; in which case the algorithm's total runtime t is updated and the control of the algorithm is transferred to the outer loop of the VNS.

The outer loop is the stopping criterion of the VNS. It compares t to the maximum runtime allowed T. VNS stops when t exceeds T.

The iterative step generates a solution $\mathbf{u}'$ from a neighborhood $N_k(\mathbf{u})$, $k = 1, \ldots, \bar{k}$, of $\mathbf{u}$ using a shaking procedure. For $k = 1$, this procedure translates one randomly chosen sphere by δ^x, δ^y and δ^z in the x, y and z directions, respectively.

For $1 < k < \bar{k}$, the procedure translates the $\min\{\frac{n}{2}, \bar{k}\}$ randomly chosen spheres in the x, y and z directions, respectively. Finally, when $k = \min\{\bar{k}, n-1\}$, it moves the k closest neighbors of a randomly chosen sphere; thus inducing variations on the position of a cluster of spheres. The parameter $\bar{k}$ is set experimentally to $\min\{\frac{n}{2}, 6\}$. In all cases, the translation distances are randomly generated from the Uniform$[-\Delta, \Delta]$, where Δ is the neighborhood parameter. If a translation causes the violation of the lower (resp. upper) bound of a variable, this latter is reset at its lower (resp. upper) bound. The shaking procedure is very useful since it offers multiple starts for NLPQLP; thus, enhances its chances of identifying a (near-) global optimum.

4.4 Computational results

In this section we provide computational results obtained by our method for solving PSC problem. This section will be much smaller in size than one in chapter 3, since we use experience from extensive computational analysis from chapter 3. In other words, we use variant of the model with fixing the cube centre, assuring that all variables are nonnegative. As a nonlinear solver, we use NLPQLP [63]. Further, we use here the better initial solution method from there. We generate random initial solution for equivalent maximization of small circle problem. In order to preserve feasibility, the initial value of r is set to 0.001. The transformation to minimization problem is then performed to be in appropriate form for the model we use. For the parameter Δ we always use value 0.5. The value of VNS parameter T is again set to 6 seconds for each independent run. Each test instance is run 10 times and best values reported. The value of $\bar{k}$ is chosen as $\min\{n, 50\}$.

The results are presented at Table 4.1. Column 1 indicates the problem size

Table 4.1: Comparing the Best Local Minima to the Best Known Radii

n	$\hat{w}_B$	$\hat{w}$	$\hat{w}_P$	w^*
3	3.4141527373	3.4142137817	3.4142135624	$\underline{3.4142135624}$
4	3.4141480640	3.4142137817	3.4142135624	$\underline{3.4142135624}$
5	3.7887647862	3.7888547050	3.7888543820	$\underline{3.7888543820}$
6	3.8855478230	3.8856182776	3.8856180832	$\underline{3.8856180832}$
7	3.9977338382	3.9978227857	3.9978227238	$\underline{3.9978227238}$
8	3.9999345267	4.0000000000	4.0000000000	4.0000000000
9	4.3093402273	4.3094012173	4.3094010768	$\underline{4.3094010768}$
10	4.6666081946	4.6666669778	4.6666666667	$\underline{4.6666666667}$
11	4.8159601052	$\underline{4.8164394712}$	4.8164397406	4.8164397406
12	4.8280686639	4.8284278518	4.8284270236	$\underline{4.8284270519}$
13	4.8282131208	4.8284278518	4.8284271247	$\underline{4.8284271247}$
14	4.8282912625	4.8284278518	4.8284271247	$\underline{4.8284271247}$
15	5.1997445225	$\underline{5.1999997920}$	5.2000000000	5.2000000000
16	5.2964155013	5.2967013468	5.2967008293	$\underline{5.2967008293}$
17	5.2996749577	$\underline{5.2998313329}$	5.2998316455	5.2998316455
18	5.3279843248	5.3282026294	5.3282011774	$\underline{5.3282011774}$
19	5.4586383844	5.4589562912	5.4589532170	$\underline{5.4589532170}$
20	5.6048721029	$\underline{5.6051549489}$	5.6051549576	5.6051549576
21	5.6431452803	$\underline{5.6427342320}$	5.6427344101	5.6430934602
22	5.7710452688	5.7712362738	5.7712361663	$\underline{5.7712361663}$
23	5.8199237336	5.8201577088	5.8201531911	$\underline{5.8201531911}$
24	5.8633943713	5.8637080741	5.8637033052	$\underline{5.8637033052}$
25	5.9589662351	5.9593423866	5.9593308221	$\underline{5.9593308221}$
26	5.9952148260	$\underline{5.9914286622}$	5.9914262468	5.9955822990
27	5.9998355203	6.0000006000	6.0000000000	6.0000000000
28	6.2421317695	6.2425479584	6.2425477277	$\underline{6.2425477277}$
29	6.2423824504	6.2426414863	6.2426406871	$\underline{6.2426406871}$
30	6.2424721518	6.2426414863	6.2426406871	$\underline{6.2426406871}$
31	6.2425174557	6.2426414863	6.2426406871	$\underline{6.2426406871}$
32	6.2425285827	6.2426414863	6.2426406871	$\underline{6.2426406871}$
33	6.4689813902	6.4680814732	6.4680780465	$\underline{6.4680780465}$
34	6.5735658875	6.5738915268	6.5738831658	$\underline{6.5738831658}$
35	6.5931612342	6.5933290674	6.5933259094	$\underline{6.5933259094}$
36	6.6970889746	6.6975333654	6.6944181198	$\underline{6.6975126034}$
37	6.7083709308	6.7086358928	6.7086344826	$\underline{6.7086344826}$
38	6.7093947495	6.7096644329	6.7096635745	$\underline{6.7096635745}$
39	6.7739983433	6.7742714525	6.7742701172	$\underline{6.7742701172}$
40	6.7998570644	6.8000010880	6.8000000000	$\underline{6.8000000000}$
41	6.9039667243	6.9042869063	6.9042712513	$\underline{6.9042712513}$
42	6.9906644142	6.9909219383	6.9907863116	$\underline{6.9907863116}$
43	7.0610542561	7.0595743359	7.0595386299	$\underline{7.0595386299}$
44	7.0991542522	7.0992323600	7.0992166891	$\underline{7.0992166891}$
45	7.1269867567	$\underline{7.1107596816}$	7.1107603214	7.1107603214
46	7.1396025069	7.1302925488	7.1302839201	$\underline{7.1302839201}$
47	7.1447631846	$\underline{7.1449572499}$	7.1449567477	7.1449574116
48	7.2254788705	$\underline{7.1449572499}$	7.1449575543	7.1449575543
49	7.3396050716	$\underline{7.3299434788}$	7.3299402906	7.3312658497
50	7.3606467872	$\underline{7.3554452729}$	7.3554013072	7.3598400994
Average	**5.8321665039**	**5.8296846262**	**5.8296132074**	**5.8298918455**
51		$\underline{7.4070535149}$	7.4061413497	7.4263293310
52		$\underline{7.4727301395}$	7.4727224395	7.4729829434
53		$\underline{7.5058057407}$	7.5057697640	7.5561707846
54		$\underline{7.5967666642}$	7.5641117363	7.6046039060
55	7.6497254384	7.6402978341	7.6207736668	$\underline{7.6392010852}$

n. Columns 2 , 3 and 4 display the side length $\hat{w}_B$, $\hat{w}$ and $\hat{w}_P$ obtained by [13], [4] and [5], respectively. Column 5 reports the side length w^* obtained by VNS. The configurations corresponding to $\hat{w}_B$ are not necessarily feasible. They may have overlapping spheres or spheres that are not totally contained within the cube. Thus, $\hat{w}_B$ will not be used for comparison purposes. For each instance of Table 4.1, the underlined value indicates the tightest upper bound. In column 4 we give the currently best known solutions from web site Packomania. However, we did not compare our results with those from Packomania, since they are obtained by many different people and many different methods. Thus, although we report results obtained by 3 different sources, we compare ours only with those from [4].

The following observations one can get from the results of Table 4.1: (i) VNS improves the best solutions obtained by [4] in 35 out of 48 instances; it represents 73% of the cases. (ii) The best improvement of an existing upper bound is 0.0010967489 occurring for $n = 55$, with the improvement averaging 0.0000391436 over the 35 improved instances. These improvements are important despite their seemingly small magnitude. (iii) Results obtained by [4] are of better quality for large values of n. This clearly indicates that another parameter values for our VNS should be chosen. That could be a task for the future work.

4.5 Conclusion

In this chapter we apply Variable neighbourhood search (VNS) approach for solving sphere packing problem within smallest containing cube, in 3D. VNS is framework for building heuristics. It starts from initial solution and use different neighbourhoods of that solution in order to improve it. The perturbation or shaking phase of the current solution is obtained by moving k ($k = 1, \ldots, \bar{k}$) sphere centres for the Δ (a parameter) in each dimension. As a local search it is used well-known software for solving nonlinear convex problems [63]. It appears that 35 out of 48 better results are reported when compared with current state of the art. Future work may contain application of this approach to other packing problems in different containers. In addition, better parameter estimation for large values of n could be performed.

Chapter 5

Conclusion

5.1 Summary

The Variable Neighborhood Search (VNS) is one of the most efficient general search frameworks. It has several heuristic variations, and can easily be adapted to continuous and discrete combinatorial problems. This thesis presents a tutorial along with a detailed literature review on recent applications of the VNS and its variants. As for other meta-heuristics, the hybridization of the VNS with other approximate or exact search algorithms enhances its efficiency and efficacy. This thesis applies a hybrid VNS to approximately solve the problem of three-dimensional circular packing, where the containing object is spherical or cubical and the items are unit spheres. This problem is relevant to many real-life applications. A successful application of a VNS-based heuristic requires a good definition of the representation of the solution, its neighbourhoods, its moves within a neighbourhood, and its local search. In this context, the neighbourhood has a special structure since it is continuous but disconnected. The proposed VNS implementation represents a solution by the three-dimensional coordinates of its solutions and its solution value by the radius of the containing sphere or by the length of the side of the containing cube. It generates a neighbouring solution by shaking one or more spheres (depending on the neighbourhood size). It applies a local search by applying the non-linear programming search technique that had obtained a large percentage of best near-global optima for other classes of complex problems while ensuring high precision. The proposed heuristic is restarted a fixed number of times so that it benefits from the diversification induced by different initial starting points. The computational results provide computational proof of the efficiency and efficacy of the VNS-based heuristic. When the containing object is a sphere, our best method is able to improve 60.4% of best known solutions and matches all other results. When the containing object is a cube, it improves 76.4% of existing solutions. Many of these solutions are suspected to be optimal, and any improvement is due a large computational precision. Indeed, the proposed approach has a higher precision level than most of the state-of-the-art approaches.

5.2 Future research

Future research might include applying the VNS to the simpler two-dimensional shapes or to more complex p-dimensional containers such as rectangular, triangular, pyramidal, and strip-shaped. Different variants of the problem may require different neighbourhood structures and/or different moves. Future and undergoing research concerns the augmentation and reduction of the problem via linearization and reformulation techniques. In real life packing of spheres, optimizes space usage is the most significant goal; yet, different issues are likewise considered. For example, cargo steadiness, multi-drop loads or weight circulations are also pertinent and should be considered during the packing or loading to ensure cargo security. Thus, adding these constraints into the three-dimensional packing problem is imperative for enhancing the applicability of this research. Such extensions could focus on the efficiency of space usage, and mixing different types of items to fill these voids. Finally, the proposed VNS heuristic could inspire the development of efficient computational heuristics for continuous optimization in other areas such as engineering and econometric.

Bibliography

[1] Hugo Pfoertner, Dense Packings of Equal Spheres in a Larger Sphere, available at www.randomwalk.de/sphere/insphr/sequences.txt, accessed on February 2011.

[2] Hugo Pfoertner, Dense Packings of Equal Spheres in a Larger Sphere, available at www.randomwalk.de/sphere/insphr/spisbest.txt, last accessed on February 2011.

[3] Jerry Donovan, Boris Lubachevsky, and Ron Graham, Optimal Packing Of Circles And Spheres, available at home.comcast.net/ davejanelle/packing.html, accessed on February 2011.

[4] Hugo Pfoertner, Densest Packings of Equal Spheres in a Cube, available at www.randomwalk.de/sphere/incube/spicbest.txt, last accessed on February 2011.

[5] Eckhardt Specht, Packing equal spheres in a cube, available at www.packomania.com, last accessed on August 2012

[6] Smart Wi-Fi for Warehouses, Ruckus Wireless, available at http://c541678.r78.cf2.rackcdn.com/brochures/`brochure_warehouse`.pdf.

[7] A. Costa, P. Hansen, and L. Libert, "Bound constraints for Point Packing in a Square". *International Conference on Operations Research and Enterprise Systems*, (2012), pp. 5–10.

[8] A. Costa, P. Hansen, and L. Libert,"On the impact of symmetry-breaking constraints on spatial Branch-and-Bound for circle packing in a square". *Discrete Applied Mathematics*, (2013),161, pp. 96–106.

[9] A. Grosso, A. R. M. J. U.Jamali, and M. Locatelli, "Finding maximin latin hypercube designs by Iterated Local Search heuristics". *European Journal of Operational Research*, (2009),197, pp. 541–547.

[10] E. G. Baum, "Toward practical 'neural' computation for combinatorial optimization problems," *Neural Networks for Computing,*American Institute of Physics, (1986).

[11] K. Bezdek "Sphere packings revisited."*European Journal of Combinatorics* (2006); 27; pp. 864-883.

[12] E. G. Birgin, and J. M. Gentil, "New and improved results for packing identical unitary radius circles within triangles," *Computers & Operations Research,*(2010), 37, pp. 1318–1327.

[13] E. G. Birgin, and F. N. C. Sobral, "Minimizing the object dimensions in circle and sphere packing problems," *Computers & Operations Research*, (2008), 35, pp. 2357–2375.

[14] M. Chiarandini, and T. Stutzle, "An application of iterated local search to the graph coloring problem," *Electronic Notes in Discrete Mathematics*, (2006),availabled at http://www.cs.colostate.edu/ howe/cs640/papers/stutzle.pdf, accessed on June 2002.

[15] J.H. Conway, and N.J.A. Sloane, "*Sphere Packings, Lattices, and Groups,*" Springer-Verlag, New York, (1999).

[16] A. Costa, and L. Libert, "Formulation symmetries in circle packing". *Electronic Notes in Discrete Mathematics*, (2010), 36, pp. 1303–1310.

[17] A. Costa, and I. Tseveendorj, "Symmetry breaking constraints for the problem of packing equal circles in a sqaure". *The Cologne-Twente Workshop on Graphs and Combinatorial Optimization*, (2011), pp. 126–129.

[18] E. Burke, T. Curtois, M. Hyde, G. Kendall, G. Ochoa, S. Petrovic, J. Vazquez-Rodrguez, and M. Gendreau, "Iterated local search vs. hyperheuristics: towards general-purpose search algorithms," *Evolutionary Computation (CEC)*, (2010), Issue No. 978-1-4244-6909-3, pp. 1-8.

[19] F. Glover, and G. Kochenberger, *Handbook of Metaheuristics*, Glover F. and Kochenberger G. Kluwer, (2003).

[20] G. Kandavanam, D. Botvich, S. Balasubramaniam, and B. Jennings, "A hybrid genetic algorithm/variable neighborhood search approach to maximizing residual bandwidth of Links for route planning," *TSSG, Waterford Institute of Technology, Ireland*, P. Collet et al. (Eds.): EA 2009, LNCS 5975, (2010), pp. 49–60.

[21] G. Yaskov, Y. Stoyan, and A. Chugay, "Packing identical spheres into a cylindrical domain," Proceedings of the Workshop on Cutting Stock Problems 2005 (WSCSP2005, September, 15-18 2005, Miercurea-Ciuc, Romania), Alutus, Miercurea-Ciuc, Romania, pp. 75-82, (2006).

[22] T. Gensane, "Dense packings of equal spheres in a cube," *Electronic Journal of Combinatorics*, (2004), 11, no. 1, R 33.

[23] H. R. Lourenco, O. Martin, and T. Stuetzle, "Iterated local search," *Glover F. and Kochenberger G. (eds.), Handbook of Metaheuristics, Kluwer*, (2003), pp. 321–353.

[24] T. C. Hales, "A proof of the Kepler Conjecture,"*Annals of Mathematics*, (2005), 162(1), pp. 1065-1185.

[25] P. Hansen and N. Mladenovic, "An introduction to variable neighborhood search," *S. Voss et al. eds., Metaheuristics, Advances and Trends in Local Search Paradigms for Optimization, Kluwer*, (1999), pp. 433–458.

[26] P. Hansen , and N. Mladenovic, "Developments of variable neighborhood search". In: Ribeiro C, Hansen P, *editors. Essays and Surveys in Metaheuristics*. Boston, Dordrecht, London: Kluwer Academic Publishers; (2001), pp. 415–440.

[27] P. Hansen , and N. Mladenovic, "Variable neighborhood search: principles and applications". *European Journal of Operational Research*, (2001);130: pp. 449–467.

[28] P. Hansen , and N. Mladenovic, "Variable neighborhood search". In: Glover F,Kochenberger G, *Handbook of metaheuristics*. Boston, Dordrecht, London: Kluwer Academic Publisher; (2003), pp. 145–184.

[29] P. Hansen , and N. Mladenovic, "Variable neighborhood search". In: E. K. Burke, and G. Kendall (Eds), *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques.*Springer; (2005), pp. 211–238.

[30] P. Hansen , and N. Mladenovic, "Variable neighborhood search methods". *Les Cahiers du GERAD*, G-2007-52,(2007).

[31] P. Hansen, and N. Mladenovic, "A Tutorial Variable Neighborhood Search ". *Les Cahiers du GERAD*, G-2003-46,(2003).

[32] M. Hifi, and R. M'Hallah, "A Literature Review of Circle and Sphere Packing Problems: Models and Methodologies," *Advances in Operations Research*, (2009) , Article ID 150624, 22 pages, doi:10.1155/2009/150624.

[33] M. Hurtgen, and J. C. Maun, "Optimal PMU placement using Iterated Local Search". *Electrical Power and Energy Systems*, (2010), 32, pp. 857–860.

[34] J. Brimberg , P. Hansen , N. Mladenovic and E. Taillard, "Improvements and Comparison of Heuristics for solving the Multisource Weber Problem". In: *Operations Research*, (2000), 48, pp. 444-460.

[35] J. Kratica, M. Leitner, and I. Ljubi, "Variable Neighborhood Search for Solving the Balanced Location Problem". In: *TECHNISCHE UNIVERSITAT WIEN, Institut fur Computergraphik und Algorithmen*, (2012),TR-186-1-12-01.

[36] J. Lazic, S. Hanafi, N. Mladenovic, and D. Urosevic, "Variable neighbourhood decomposition search for 0-1 mixed integer programs". *Computers Operations Research* , (2010), 37, pp. 1055–1067.

[37] J. F. Liu, Y. L. Yao, Y. Zheng, H. T. Geng, and G. C. Zhou, "An effective hybrid algorithm for the circles and spheres packing problems."In: D. Z. Du, X. D. Hu, P. M. Pardalos (Eds), *Combinatorial Optimization and Applications: Third International Conference*. Berlin: Springer; (2009), pp. 135-144.

[38] J. Moreno-Perez, P. Hansen , and N. Mladenovic, "Parallel variable neighborhood search ". *Les Cahiers du GERAD*, G-2004-92,(2004).

[39] L. F. Garcia, B. Melian Batista, J. A. Moreno Perez, and J. M. Moreno Vega, "The parallel variable neighbourhood search for the P-median problem". *Journal of Heuristics*, (2002), 8, pp. 375–388.

[40] M. Hifi, and R. M'Hallah, "Beam search and non-linear programming tools for the circular packing problem". *International Journal of Mathematics in Operational Research*, (2009), 1, pp. 476–503.

[41] M. Hifi, and R. M'Hallah, "A dynamic adaptive local search based algorithm for the circular packing problem". *European Journal of Operational Research* , (2007), 183, pp. 1280–1294.

[42] M. Gan, N. Gopinathan, X. Jia, and R. A. Williams, "Predicting packing characteristics of particles of arbitrary shapes," *KONA Powder & Particle Journal*, (2004), 22, pp. 82-93.

[43] M. Maric, Z. Stanimirovic, and N. Milenkovic, "Metaheuristic methods for solving the Bilevel Uncapacitated Facility Location Problem with Clients' Preferences". *In Faculty of Mathematics, University of Belgrade, Serbia,Electric Notes in Discrete Mathematics*, (2012), 39, pp. 43–50.

[44] N. Mladenovic, "A variable neighborhood algorithm-A new metaheuristic for combinatorial optimization". *Presented at Optimization Days, Montreal* , (1995), p. 112.

[45] N. Mladenovic , and P. Hansen, "Variable neighborhood search.' *Computers and Operations Research*, (1997);24:1097–1100.

[46] N. Mladenovic, R. Todosijevic, and D. Urosevic, "An efficient general variable neighborhood search for large travelling salesman problem with time windows", *Yugoslav Journal of Operations Research*, (2012), 22, no. 2.

[47] N. Mladenovic, J. Petrovic, V. Kovacevic-Vujcic, and M. Cangalovic "Solving spread spectrum radar polyphase code design problem by tabu search and variable neighbourhood search"*European Journal of Operational Research*, (2003), 151, pp. 389–399.

[48] N. Nikolic, Igor Grujicic, and Dorde Dugosija, "Variable neighborhood descent heuristic for covering design problem"*Electronic Notes in Discrete Mathematics*, (2012), 39, pp. 193–200.

[49] P. R. J. Ostergard "Book review," *Computers and Operations Research*, (2008), 36, pp. 276–278.

[50] P. Hansen, C. Oguz, and N. Mladenovic, "Variable neighborhood search for minimum cost berth allocation," *European Journal of Operational Research*, (2008), 191, pp. 636–649.

[51] P. Hansen, N. Mladenovic, and J. Moreno-Perez, "Variable neighbourhood search: methods and applications," *INVITED SURVEY, 4OR*, (2008), 6, pp. 319–360.

[52] P. Hansen, N. Mladenovic, and J. Moreno-Perez, "Variable neighbourhood search: methods and applications," *Annals of Operations Research*, (2010), 175, pp. 367–407.

[53] P. Hansen, N. Mladenovic, and D. Perez-Brito, "Variable neighborhood decomposition search" *Journal of Heuristics*, (2001), 7, pp. 335–350.

[54] P. Hansen, N. Mladenovic, and D. Urosevic "Variable neighborhood search and local branching," *Computers & Operations Research*, (2006), 33, pp. 3034–3045.

[55] R. M'Hallah, and A. Alkandari , "Packing unit spheres into a cube using VNS," *Electronic Notes in Discrete Mathematics*, (2012), 39, pp. 201–208.

[56] R. M'Hallah, A. Alkandari, and N. Mladenovic, "Packing unit spheres into the smallest sphere using VNS and NLP," *Computers Operations Research*, (2013), 40, pp. 603–615.

[57] R. Qu, Y. Xu, and G. Kendall, "A Variable Neighborhood Descent Search Algorithm for Delay-Constrained Least-Cost Multicast Routing, In T. Stutzle (Ed.)," *LION 3, LNCS 5851*, Springer-Verlag, Berlin, Heidelberg, (2009), pp. 15–29.

[58] C. R. Reeves, "Modern heuristic techniques for combinatorial problems," *Blackwell Scientific Press*, (1993).

[59] S. Brito, G. Fonseca, T. Toffolo, H. Santos, and M. Souza, "A SA-VNS approach for the High School Timetabling Problem," *Electronic Notes in Discrete Mathematics*, (2012), 39, pp. 169-176.

[60] S. Hanafi, J. Lazic, and N. Mladenovic, "Variable neighbourhood pump heuristic for 0–1 mixed integer programming feasibility," *Electronic Notes in Discrete Mathematics*, (2010), 36, pp. 759–766.

[61] S. Kucherenkoa , P. Belottib , L. Libertic, and N. Maculand , "New formulations for the Kissing Number Problem", *Discrete Applied Mathematics*,(2007), 155, pp. 1837–1841.

[62] J. Sanchez-Oro, and A. Duarte, “An experimental comparison of Variable Neighborhood Search variants for the minimization of the vertex-cut in layout problems”. *Electronic Notes in Discrete Mathematics*,(2012), 39,pp. 59–66.

[63] K. Schittkowski , “NLPQLP: A Fortran Implementation of a Sequential Quadratic Programming Algorithm with Distributed and Non-Monotone Line Search - User’s Guide,”Report, Department of Computer Science, University of Bayreuth, (2010), V 3.1.

[64] R. Silva, and S. Urrutia, “A general VNS heuristic for the traveling salesman problem with time windows ,” *Discrete Optimization*, (2010), 7, pp. 203–211.

[65] Y. G. Stoyan, “Mathematical methods for geometric design,” in: *Advances in CAD/CAM: Proceedings of PROLAMAT82*, Leningrad, USSR, 16–18 May 1982, pp. 67–86. North-Holland, Amsterdam, (2003).

[66] Y. G. Stoyan, and G. Yaskov, “Packing Identical Spheres into a Rectangular Parallelepiped,” In: A. Bortfeldt, J. Homberger, H. Kopfer, G. Pankratz, R. Strangmeier, *Intelligent Decision Support. Current Challenges and Approaches*, Betriebswirtschaftlicher Verlar Dr. Th. Gabler, GWV Fachverlage GMbH, Wiesbaden, (2008), pp. 47-67.

[67] Y. G. Stoyan and G. N. Yaskov, “A mathematical model and a solution method for the problem of placing various sized circles into a strip,” *European Journal of Operational Research,* (2004), 156, pp. 590-600.

[68] Y. G. Stoyan, and G. N. Yaskov, “Packing identical spheres into a right circular cylinder,” Proceedings of the 5th ESICUP Meeting, L’Aquila, Italy, April 20 - 22, (2008).

[69] Y. G. Stoyan, and G. N. Yaskov “Packing congruent hyperspheres into a hypersphere,”*Journal of Global Optimization,* (2012), 52, pp. 855–868.

[70] A. Sutou, and Y. Dai, “Global optimization approach to unequal sphere packing problems in 3D,”*Journal of Optimization Theory and Applications,* (2002), 114, No. 3, pp. 8671–694.

[71] Szabo, P. G., M. C. Markot, T. Csendes, E. Specht, L. G. Casado, and I. Garcia, *New Approaches to circle Packing in a Square*, Springer,New York (2007).

[72] T. G. Crainic, M. Gendreau, P. Hansen, and N. Mladenovic, “Cooperative parallel variable neighborhood search for the p -median,”*Journal of Heuristics,* (2004), 10, pp. 293–314.

[73] M. Toksari, and E. Guner, “Solving the unconstrained optimization problem by a variable neighborhood search,”*Journal of Mathematical Analysis and Applications,* (2007), 328, pp. 1178–1187.

[74] S. Torquato, and F. H. Stillinger, "Exactly solvable disordered sphere-packing model in arbitrary-dimensional Euclidean spaces,"*Physical Review,* (2006), E 73, pp. 031106–031114.

[75] J. Wang, "Packing of unequal spheres and automated radiosurgical treatment planning," *Journal of Combinatorial Optimization,*(1999), 3, pp. 453-463.

[76] S. R. Williams, and A. P. Philipse, "Random packing of spheres and sphero-cylinders simulated by mechanical contraction," *Physics Review*, (2003), E 67, pp. 051301-051309.

[77] Y. G. Stoyan, G. N. Yaskov, and G. Scheithauer, "Packing of various radii solid spheres into a parallelepiped," *Central European Journal of Operational Research*, (2003), 11, pp. 389-407.

[78] Y. G. Stoyan, J. Terno, G. Scheithauer, and T. Romanova, "Φ functions for primary 2D-objects," *Studia Informatica Universalis, International Journal on Informatics*, Special Issue on Cutting, Packing and Knapsacking,(2002), 2, pp. 1-32.

[79] G. Wascher, H. Haussner, and H. Schumann, "An improved typology of cutting and packing problems," *European Journal of Operational Research, Special Issue on Cutting and Packing,*(2007), 183, no. 3, pp. 1109-1130.

[80] G. Zoutendijk, "Nonlinear Programming, Computational Methods, Integer and Nonlinear Programming," *North Holland Publishing Co*, Amsterdam,(1970).