HEURISTIC APPROACHES FOR MAXIMIN DISTANCE AND PACKING PROBLEMS

by

Abdur Rakib Muhammad Jalal Uddin Jamali
XXI Cycle

A thesis submitted for the partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

Supervisor: ***Prof. Marco Locatelli***

Co-supervisor: ***Dr. Andrea Grosso***

Dipartimento di Informatica
Università degli Studi di Torino
Torino, Italia.

Date: 18 - 02 - 2009.

UNIVERSITA' DEGLI STUDI DI TORINO

DIPARTIMENTO DI INFORMATICA

DOTTORATO DI RICERCA IN INFORMATICA

CICLO XXI

TITOLO DELLA TESI:

Heuristic Approaches for Maximin Distance and Packing Problems

TESI PRESENTATA DA:

Abdur Rakib Muhammad Jalal Uddin Jamali

TUTOR:

Prof. Marco Locatelli

CO-ADVISOR:

Dr. Andrea Grosso

COORDINATORE DEL CICLO:

Prof. Pietro Torasso

ANNI ACCADEMICI: *2005-2008*

SETTORE SCIENTIFICO-DISCIPLINARE DI AFFERENZA: *INF/01*

Abstract

In this thesis we mainly deal with two problems – experimental design and packing problems. In the field of experimental design problems we consider maximin Latin Hypercube Designs (LHDs). In the field of packing problems we consider those of packing n equal or unequal circles in a circular container with minimum radius. Both problems can be formulated as optimization ones. The former is a combinatorial problem, while the latter is a continuous one.

We propose heuristic approaches to tackle these problems. These are Iterated Local Search (ILS) heuristics for maximin LHDs, and Basin Hopping (BH) heuristics for packing problems. Actually, ILS and BH approaches have strong similarities and could be described within an unified framework. However, following the literature, where ILS approaches are mainly applied to combinatorial problems, while BH approaches are mainly applied to continuous problems, we will keep them apart.

In order to deal with maximin LHDs, we propose two ILS variants, corresponding to two distinct optimality criteria which are employed to drive the search among LHDs. Extensive experiments are performed for the investigation of the strengths and weaknesses of the algorithms. A remarkable finding is that the most efficient method, though time consuming, performs a non monotonic search, driven by an appropriate objective function, within the space of LHDs. The proposed approaches are extensively compared with the existing ones in the literature, and many improved results with respect to best known ones, are obtained. In particular, the proposed methods seem to outperform the existing ones when the dimension of the design points increases. Finally, we also discuss about the time complexity of the algorithms: by mixing theoretical results with experimental ones, we derive an empirical formula for each ILS variant, returning the expected run time as a function of the number of design points and of their dimension.

To deal with the problem of packing *equal* circles in a circular container with minimum radius, we propose a variant of BH, namely Monotonic BH (MBH) and its population based counterpart, Population BH (PBH). Extensive computational experiments are performed both to analyze the problem at hand, and to choose in an appropriate way the parameter values for the proposed methods. Different improvements with respect to the best results reported in the literature are detected. The problem of packing *unequal* circles in a circular container with minimum radius is also attacked with the MBH and PBH approaches, but some components of these approaches are adapted in order to fully exploit the peculiarities of the problem with unequal circles (in particular, its combinatorial nature due to the different radii of the circles). Again extensive computational experiments are performed and improvements with respect to the existing literature are detected.

Dedication

To

my wife Nasima Perveen Sathee
&
Brother-in-law Mashiur Rahman

Acknowledgment

After thanking to the Almighty, I would like to express my greatest appreciation toward my two thesis supervisors, Prof. Marco Locatelli and Dr. Andrea Grosso, for their continuous guidance, support and friendship. My sincere gratitude goes to Prof. Locatelli as well as Dr. Grosso for accepting me as a PhD student into their research group. They keeping constant faith in me. Without their continuous encouragement, I would still be in my PhD dreams. Prof. Locatelli has a lot of research experience in this area. He has been a great source of ideas, knowledge and feedback for me. They are great mentors and I learned so much from them through this long academic journey.

I wish to express my profound gratitude to Chair of the PhD program Prof. Pietro Torasso and all of the other members of the committee for accepting me into the PhD program. Thanks to the University of Turin for making my PhD studies there possible with the grand of the scholarships. Its outstanding learning environment helped lay a solid academic foundation. I would like to thanks Dott.ssa Alessandra Pachi, officer of ISASUT, Claudio Schifanella, short term researcher, and Mariuccia Furci, officer of Collegio Einaudi who helped me whenever I needed assistance regarding administrative as well as other personal problems for staying abroad. I am also thankful to all members of the Department of Computer Science as well as technical and nontechnical staff for their assistance during my research work.

My thesis' referees provided invaluable feedback and advice on my thesis. I really appreciate their time and help. I wish to convey my thanks to Prof. F. Schoen, Univeristy of Firenze, Italy, who provided useful software for packing problems. I would also convey my gratefulness to the people mananiging the web sites `www.spacefillingdesigns.nl` and `www.packomania.com`, which provided us valuable information regarding our research works.

I heartily express my gratefulness to authorities of the Khulna University of Engineering & Technology (KUET) for granting me to study abroad. I am specially grateful to Prof. Fouzia Rahman, Prof. Dr. M. Arif Hossain, Prof. Dr. M. M. Hashem, M. Tauhiduzzaman, and M. Nuruzzaman, KUET, Bangladesh, for their continuous inspiration and administrative assistance during my study abroad.

I give heartfelt thanks to all of my family members as well as my wife's family members. Specially I am indebted to my wife Nasima Perveen Sathee for her continuous encouragement and understanding when we are living alone too many months as for staying abroad. She had to look after our two beautiful children Tahmid Kawser Washee and Ahanab Tajwar Tur all the three years. I am really obliged to express my heartiest thanks to her. Finally, I would like to express my beloved thanks to my nice kids, for their sacrifice to get affectionate and love what they ought to deserve, during my research period.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

In an optimization problem we search for the (global) minimum (maximum) of a function over a given domain; we can formulate it as follows

$$\min \text{ (or } \max) \{f(X) : X \in D \subseteq \mathbb{R}^n\} \tag{1.1}$$

where f is called *objective function* and maps $\mathbb{R}^n$ to $\mathbb{R}$, and D is called *feasible set*, usually defined by a set of equality and/or inequality constraints. The problem is called *non-convex* if f is not a convex function (or concave in maximization problems) and/or D is not a convex set. The problem is said to be *unconstrained* if $D = \mathbb{R}^n$; when $D \subset \mathbb{R}^n$, we are dealing with a *constrained* optimization problem. If D is a discrete set, then we call the optimization problem a *combinatorial* one. If D is a continuous set we call the optimization problem a *continuous* one.

Important definitions are those of local and global optimizers. A global minimizer (maximizer) is simply the best point within the feasible region, i.e., the one with lowest (largest) objective function value if we are minimizing (maximizing). More formally, X^* is a global minimizer of f over D if

$$f(X^*) \leq f(X) \; \forall \; X \in D \tag{1.2}$$

In order to define local minimizers we need to introduce the concept of neighborhood. For continuous problems the natural choice is to consider as a neighborhood, $\mathcal{N}_\eta$, for some $X \in D$ a n-dimensional sphere centered at X and with some positive radius η , i.e.

$$\mathcal{N}_\eta(X) = \{Y : \|Y - X\|_2 \leq \eta\} \tag{1.3}$$

where $\|\cdot\|_2$ denotes the Euclidean distance. Then, in the continuous case we define X' a local minimizer, if for some $\eta > 0$, X' is the best point within $\mathcal{N}_\eta(X') \cap D$, i.e.,

$$f(X') \leq f(X) \; \forall \; X \in \mathcal{N}_\eta(X') \cap D \tag{1.4}$$

For combinatorial problems the definition of a neighborhood is less obvious and often problem dependent. Anyway, once a neighborhood structure $\mathcal{N}_c$ has been defined, the definition of a local minimizer is completely analogous to (1.4) with $\mathcal{N}_\eta$ replaced by $\mathcal{N}_c$. Note that any global minimizer is also a local minimizer (independently from the neighborhood). Also note that while efficient procedures for the detection of local minimizers are usually available, the task of finding a global minimizer is usually a much more challenging one. Exact methods exist, including branch-and-bound techniques, cutting algorithms, branch-and-cut algorithms, dynamic programming. However, often only heuristic approaches are

possible because returning a certificate of optimality would require unacceptably long computation times. The efficiency of these heuristic procedures often depends on an appropriate combination of general approaches (i.e., approaches which can be applied to general optimization problems) with specific problem knowledge.

While there exists a huge variety of combinatorial and continuous optimization problems, in this thesis we will focus our attention on two classes of optimization problems: *maximin distance* problems and *packing* problems. In the first class we have to place a given number of points in some given area in such a way that their minimal "distance" is maximized. Within this class we will consider a combinatorial problem, maximin Latin Hypercube Design (LHD), where the area is a grid of points with integer coordinate values (which is related to *experimental design* problem), and a continuous one, where the area is a circular one. In the second class we have to place objects within some area, whose dimension is controlled by some parameter, without overlapping and in such a way that the overall dimension of the area is as small as possible. In this class we will consider two continuous problems: the problem of packing *equal* circles within a circular container with minimum radius (which, in fact, is equivalent to the second maximin distance problem introduced above, as we will discuss more thoroughly later on), and the same problem with *unequal* circles.

1.1 Statements of the problems

Maximin distance problems aim at placing N points within a region C in such a way that they satisfy some properties $\mathcal{P}_1, \ldots, \mathcal{P}_u$ and that, given some distance d between pairs of points, the minimal distance between them is maximized. Formally, the problem is the following

$$\begin{array}{ll} \max & \min_{i \neq j} \; d(x_i, x_j) \\ & x_1, \ldots, x_N \in C \\ & x_1, \ldots, x_N \text{ satisfy properties } \mathcal{P}_1, \ldots, \mathcal{P}_u \end{array} \tag{1.5}$$

In its most general formulation the packing problem can be defined as follows. Given a container which depends on a size parameter r and denoted by $C(r) \subset \mathbb{R}^d$, and given n geometrical objects whose position in the d-dimensional space depends on u position parameters $\alpha_{i1}, \ldots, \alpha_{iu}$, i.e., $D_i = D_i(\alpha_{i1}, \ldots, \alpha_{iu}) \subset \mathbb{R}^d$, $i = 1, \ldots, n$, we would like to choose the parameters in such a way that all the objects are packed into the container without overlapping (the objects can at most "touch" each other) and the size of the container is minimized. More formally, the problem is the following

$$\begin{array}{lll} \min & r & \\ & D_i(\alpha_{i1}, \ldots, \alpha_{iu}) \subseteq C(r) & i = 1, \ldots, n \\ & D_i^0(\alpha_{i1}, \ldots, \alpha_{iu}) \cap D_j^0(\alpha_{j1}, \ldots, \alpha_{ju}) = \emptyset & i \neq j \end{array} \tag{1.6}$$

where D_i^0 denotes the interior of D_i.

As pointed out previously, in the thesis we will deal with two problems belonging to each of the two classes. The first maximin distance problem is maximin

Latin Hypercube Design (LHD). In such problem with respect to (1.5) we have that

$$C = \{0, 1, \ldots, N-1\}^k$$

for some dimension k, $u = 1$ and $\mathcal{P}_1$ requires that no two points have a common coordinate value (equivalently, for each single coordinate, the values of such coordinate for the N points must represent a permutation of $0, 1, \ldots, N-1$). Being C a discrete set, the problem is a combinatorial one. The second maximin distance problem is related to placing points in a circular area, which, without loss of generality, can always be considered as the unit circle. In this case with respect to (1.5) we have that

$$C = \{(x, y) \ : \ x^2 + y^2 \leq 1\},$$

and points do not have to satisfy further properties.

In the two packing problems we have to place (equal or unequal) circles in a circular container. With respect to (1.6) we have that: the objects are equal or unequal circles of given radius; the container is a circle; $u = 2$; the position parameters α_{i1} and α_{i2} correspond to the coordinates of the center of circle i; the size parameter r is the radius of the circular container.

We remark that the problem of packing equal circles in a circular container with minimum radius, and the problem of placing points in a circular area in such a way that their minimal distance is maximized, are in fact equivalent, in the sense that given an optimal solution and the optimal value for one of the two problems, we can easily derive through simple formulas an optimal solution and the optimal value also for the other problem. For this reason, in what follows we will only discuss one of the two problems, namely the packing one.

1.2 Latin Hypercube Designs: literature review

Since physical experiments are inevitably very expensive and time consuming, computer experiments are widely used for simulating physical characteristics and for the design and development of products (for examples, see [67]). A computer experiment is modeled as a realization of a stochastic process, often in the presence of nonlinearity and high dimensional inputs (see Sacks, Welch, Mitchell and Wynn [203]). In order to perform efficient data analysis and prediction and in order to determine the best settings for a number of design parameters that have an impact on the response variable(s) of interest and which influence the critical quality characteristics of the product or process, it is often necessary to set a good design as well as to optimize the product or process design. In computer experiments, instead of physically doing an experiment on the product, *mathematical models* describing the performance of the product are developed using engineering/physics laws. Then the mathematical models are solved on computers through numerical methods such as the finite element method. A computer simulation of the mathematical models is usually time-consuming and there is a great variety of possible input combinations. For these reasons, *meta-models* [13, 217] that model the quality characteristics as explicit functions of the design parameters are constructed. Such a meta-model, also

called a (global) approximation model or surrogate model, is obtained by simulating a number of design points. Since a meta-model evaluation is much faster than a simulation run, in practice such a meta-model is used, instead of the simulation model, to gain insight into the characteristics of the product or process and to optimize it. Therefore, a careful choice of the design points at which performing simulations in order to build the meta-model, is of primary importance.

As is recognized by several authors, the choice of the design points for computer experiments should at least fulfill two requirements (for details see Johnson et al. [125] and Morris and Mitchell [178]). First of all, the design should be *space-filling* in some sense. When no details on the functional behavior of the response parameters are available, it is important to be able to obtain information from the entire design space. Therefore, design points should be *evenly spread* over the entire region. Secondly, the design should be *non-collapsing*. When one of the design parameters has (almost) no influence on the function value, two design points that differ only in this parameter will *collapse*, i.e., they can be considered as the same point that is evaluated twice. For deterministic functions this is not a desirable situation. Therefore, two design points should not share any coordinate value when it is not known a priori which parameters are important.

The latter requirement is fulfilled by employing Latin Hypercube Designs (LHDs). Such designs, proposed by Beckman, Conover and McKay [169], are evenly distributed in each one-dimensional projection and are thus *non-collapsing*. Unfortunately, randomly generated LHDs almost always show poor *space-filling* properties. On the other hand, maximin distance designs, proposed by Johnson, Moore and Ylvisaker [125], have very good *space-filling* properties but often no good projection properties under the Euclidean or the Rectangular distance. To overcome this shortcoming, Morris and Mitchell [178] suggested to search for maximin LHDs when looking for "optimal" designs. Although the search for maximin LHDs will be one of the problems discussed in this thesis, it is important to point out that also other definitions of "optimality" for designs exist in the literature. These are not discussed in detail throughout the thesis (we refer, e.g., to the book of Santner et al. [205]), but, for the sake of completeness, in the following literature review we will mention some of them, together with a short discussion of the methods employed to return "optimal" (according to the selected definition) designs.

In [128] F. Jurecka et al. the concept of robust design is presented and the need for meta-models within this framework is elaborated. They also introduced a method to sequentially update the meta-models during the robust design optimization process through strategies typically used in global optimization.

Bates et al. [15] obtain designs for computer experiments by exploring so-called lattice points and using results from number theory.

Fang et al. in [65, 66] defined a uniform design as a design that allocates experimental points uniformly scattered on the domain. Uniform designs do not require orthogonality. They consider projection uniformity over all subdimensions. In [66] they classify uniform designs as space-filling designs.

In [212] Sebastiani and Wynn considered maximum entropy sampling criterion for the optimal Bayesian experimental design. The main contribution of this paper is the extension of the MES principle for the estimation of the problems. Currin [47] also considered an entropy-based design criterion for Bayesian prediction of deterministic functions.

In [119] R. L. Iman and W. J. Conover proposed a design by minimizing a linear correlation criterion for pairwise factors. This is modified into a polynomial canonical correlation criterion by Tang [238].

Johnson et al. in [123] and Morris and Mitchell in [178] proposed the maximin distance criterion which maximizes the minimum distance between design points. Note that a maximin design is certainly space-filling, but not necessarily non-collapsing, unless the LHD requirement is imposed.

Stinstra et al. [221] proposed sequential heuristic algorithms for constrained maximin designs by considering high number of design sites with small volume of feasible design space and other constraints. They also used their methods in many practical situations.

Lee and Jung [145] proposed maximin eigenvalue sampling, that maximizes minimum eigenvalue, for Kriging model where maximin eigenvalue sampling uses eigenvalues of the correlation matrix. The Kriging model is obtained from sampled points generated by the proposed method. Note that the Kriging model [137] is used to compare the characteristics of proposed sampling design with those of maximum entropy sampling.

The maximin design problem has also been studied in location theory. In this area of research, the problem is usually referred to as the max-min facility dispersion problem (see Erkut [63]); facilities are placed such that the minimal distance to any other facility is maximal. Again, the resulting solution is certainly space-filling, but not necessarily non-collapsing.

In statistical environments Latin Hypercube sampling is often used. In such an approach, points on the grid are sampled without replacement, thereby deriving a random permutation for each dimension (see McKay et al. [169]).

Giunta et al. [77] give an overview of pseudo- and quasi-Monte Carlo sampling, Latin hypercube sampling, orthogonal array sampling, and Hammersley sequence sampling.

McKay et al. [169], Stein [218] and Owen [192] had shown that LHDs perform much better than completely randomized designs. More recently, algorithms have been used to construct systematic LHDs under various optimality criteria. A LHD always has *non-collapsing* properties but not necessarily good *space-filling* property. In particular, as already remarked, randomly generated LHDs often show poor *space-filling* properties. Therefore, the search for "optimal" LHDs has attracted attention (see, e.g., [178, 193, 237, 265, 266]). Different optimality criteria for LHDs have been proposed, including maximum entropy

designs (Shewry and Wynn, [214]; Currin et al., [47]), Integrated Mean Squared Error (IMSE) of prediction (Sacks et al. [203]) and minimax and maximin distance designs (Johnson et al. [125]).

Hongquan Xu in [263] introduced the concept of universal optimality from optimum design theory into computer experiments, and then exhibited some universally optimal designs with respect to different distance measures. He showed that Latin Hypercubes and saturated orthogonal arrays are universally optimal with respect to Hamming distance [100], and that universally optimal designs with respect to Lee distance [143] are also derived from Latin Hypercubes and saturated orthogonal arrays.

Lin in [153] proposed several methods for extending the uniform sampling to higher dimensions. The method has also been used to construct LHDs with low correlation of first-order and second-order terms. It generates orthogonal LHDs that can include many more factors than those proposed by Ye [265].

Cioppa in his dissertation [40] developed a set of experimental designs by considering orthogonal Latin hypercubes and uniform designs to create designs having near orthogonality and excellent space-filling properties. Multiple measures were used to assess the quality of candidate designs and to identify the best one.

Tang in [238] proposed a LHD by the extension of the concept of Iman and Conover in [119], namely minimizing a polynomial canonical correlation criterion for pairwise factors.

Park in [193] and Sacks in [203] constructed optimal LHDs in which IMSE and entropy optimization criteria were considered. To construct optimal LHDs, Park presented an approach based on the exchanges of several pairs of elements in two rows. His algorithm first selects some active pairs which minimize the objective criterion value by excluding that pair from the design. Then, for each chosen pair of two points i_1 and i_2, the algorithm considers all possible exchanges $x_{i_1 j_1} \leftrightarrow x_{i_2 j_1}, \ldots, x_{i_k j_1} \leftrightarrow x_{i_k j_1}$ for $k \leq l$ and find the best exchange among them.

Leary et al. [142] proposed orthogonal-array-based LHDs for obtaining better space-filing property. As an optimal criterion, they consider the sum of (square of) reverse inter-site distances.

Ye in [265] constructed orthogonal LHDs in order to enhance the utility of LHDs for regression analysis. Ye defines an Orthogonal Latin Hypercube (OLHC) as a Latin Hypercube for which every pair of columns has zero correlation. Furthermore, in Ye's OLHC construction, the element-wise square of each column has zero correlation with all other columns, and the element-wise product of every two columns has zero correlation with all other columns. These properties ensure the independence of estimates of linear effects of each variable and the estimates of the quadratic effects and bilinear interaction effects are uncorrelated with the estimates of the linear effects.

In [219] Steinberg and Lin constructed LHDs in which all main effects are or-

thogonal. Their method can also be used to construct LHDs with low correlation of first-order and second-order terms. It also generates orthogonal LHDs that can include many more factors than those proposed by Ye [265].

Morris [177] and Kleijnen [132] make it clear that many simulation models involve several hundred factors or even more. Consequently, factor screening is useful in computer experiments for reducing the dimension of the factor space before carrying out more detailed experiments. In [33] Butler proposed optimal and orthogonal LHDs which is suitable for factor screening.

Fang et al. [65] proposed threshold accepting heuristic approaches for optimal LHDs to produce low discrepancy designs compared to theoretic expectation and variance. They considered centered L_2-discrepancy for optimizing the designs.

Olsson [190] suggested Latin Hypercube sampling as a tool to improve the efficiency of different importance sampling methods for structural reliability analysis. Stocki [222] and Liefvendahl and Stocki [151] proposed probabilistic search algorithm, namely Column-wise Pair-wise (CP) search algorithms and Genetic algorithms to construct optimal LHDs. For the optimal criterion they considered energy function (the sum of the norms of the repulsive forces if the samples are considered as electrically charged particles) as proposed in [9]. To improve the reliability, Stocki [222] considered the pairwise correlation. Liefvendahl and Stocki [151] also compared the performance of the CP and genetic algorithms for optimal LHDs.

By using the Latin Hypercube sampling method in [118], Hwan Yang performed the uncertainty and sensitivity analysis for the time-dependent effects in concrete structure. The results of the Latin Hypercube simulations were used to determine which of the model parameters are most significant in affecting the uncertainty of the design [120]. For each sample, a time-dependent structural analysis was performed to produce response data, which were then analyzed statistically.

G. Wang [248] used the Latin Hypercube Design (LHD) instead of the Central Composite Designs (CCD), for improvement of Adaptive Response Surface Method (ARSM). Note that ARSM was developed to search for the global design optimum for computation-intensive design problems. Also note that Response Surface Method (RSM) plans a group of design alternatives and performs the design analysis and simulation simultaneously on these design alternatives. Then an approximation model, called a response surface, is constructed.

van Dam in [48] derives interesting results for two-dimensional minimax LHDs. Bates et al. [16] propose a permutation genetic algorithm to find optimal Audze-Eglais LHDs. Crary et al. [45] developed I-OPT^{TM}to generate LHDs with minimal IMSE.

In Santner et al. [205], Bursztyn and Steinberg [32], [215], it is shown that maximin optimal LHDs generally speaking yield the best approximations.

Jin et al. [122] proposed an enhanced stochastic evolutionary algorithm for finding maximin LHDs. They also apply their method to other space-filling criteria, namely the optimal entropy and centered L_2 discrepancy criteria.

van Dam et al. in [49] derive general formulas for two-dimensional maximin LHDs, when the distance measure is ℓ^∞ or ℓ^1, while for the ℓ^2-distance measure (approximate) maximin LHDs up to 1000 design points are obtained by using a branch-and-bound algorithm and constructing (adapted) periodic designs.

van Dam et al. in [50] proposed some bounds, for the separation distance of certain classes of maximin LHDs, which are useful for assessing the quality of approximate maximin LHDs. By using some of the special properties of LHDs, they were able to find new and tighter bounds for maximin LHDs. Besides these bounds, they presented a method to obtain a bound for three-dimensional LHDs that is better than Baer's bound for many values of N. They also constructed maximin LHDs attaining Baer's bound for infinitely many values of N in all dimensions.

Morris and Mitchell [178] adopted a simulated annealing (see, e.g., [8]) to find approximate maximin LHDs for up to five dimensions and up to 12 design points, and a few larger values, with respect to the ℓ^1- and ℓ^2-distance measure. In Morris and Mitchell's algorithm, a search begins with a randomly chosen LHD, and proceeds through examination of a sequence of designs, each generated as a perturbation of the preceding one. A perturbation D_{try} of a design D is generated by interchanging two randomly chosen elements within a randomly chosen column in D. The perturbation D_{try} replaces D if it leads to an improvement. Otherwise, it will replace D with probability $\pi = \exp[-\{\phi(D_{try}) - \phi(D)\}/t]$, where t is the preset parameter known as the "temperature "and ϕ is some measure of the quality of the design. Li and Wu [150] considered a class of Column-wise Pair-wise (CP) algorithms in the context of the construction of optimal supersaturated designs. A CP algorithm makes exchanges on the columns in a design and can be particularly useful for designs that have structure requirements on the columns. Note that each column in a LHD is a permutation of $\{0, \ldots, N-1\}$. At each step, another permutation of $\{0, \ldots, N-1\}$ is chosen to replace a column so that the LHD structure is retained.

Joseph and Hung in [127] proposed a multi-objective optimization approach to find good LHDs by combining correlation and distance performance measure. They proposed a modified simulated annealing algorithm with respect to [178]. Instead of randomly choosing a column and two elements within that column, as in [178], they choose them judiciously in order to achieve improvement in their multi-objective function.

Ye et al. [266] and Li and Ye [149] proposed an exchange algorithm for finding approximate optimal LHDs, but they consider symmetric latin hypercube designs (SLHDs). The symmetry property is used as a compromise between computing effort and design optimality. However, one important change had made to accommodate the special structure of SLHD. For a SLHD two simultaneous pair exchanges were made in each column to retain the symmetry. Ye et al. [266] considered maximin as an optimal criterion, whereas Li and Ye [149]

considered both the maximin and the entropy optimal criterion.

Husslage et al. in [116] constructed nested maximin designs in two dimensions. They showed that different types of grids should be considered when constructing nested designs and discussed how to determine which grid is the best for a specific computer experiment.

Using (adapted) periodic designs and simulated annealing, Husslage et al. in [117] extended the known results and construct approximate maximin Latin hypercube designs for up to ten dimensions and for up to 100 design points. All these designs can be downloaded from http://www.spacefillingdesigns.nl. Inspired by the paper [178], in which authors show that LHDs often have a nice periodic structure, Husslage et al. developed adapted periodic designs. By considering periodic and adapted periodic designs, approximate maximin LHDs for up to seven dimensions and for up to 100 design points are constructed. They have shown that the periodic heuristic tends to work well even for a small number N of design points at low values of the dimension k, but as k increases the periodic heuristic tends to get better than other approaches like simulated annealing only at large N values.

In the simulated annealing algorithm of Husslage et al. in [117], four different neighborhoods have been considered. In all four neighborhoods the main idea is to change two points of the current LHD by exchanging one or more coordinate values. In three of the four neighborhoods, one point is required to be a critical point (a critical point is a point which is at separation distance, i.e., at a distance equal to the minimal one, from one of the other points). In the first neighborhood, one point j_1 is selected randomly from all critical points and the other point j_2 randomly from all remaining points. This implies that the second point can either be a critical or noncritical point. Once the points are selected, the number of coordinates to change is randomly selected. Due to symmetry, at most $\lfloor k/2 \rfloor$ coordinates are changed. Subsequently, the coordinates to change are randomly selected. The values of the two points in these coordinates are then exchanged, which results in a new LHD. The second neighborhood is very similar to the first. The only difference is that always one coordinate is selected instead of a random number of coordinates. Note that for $k = 3$ the two neighborhoods are the same. In the third neighborhood, also one coordinate is changed, however, now the coordinate is not randomly selected. Instead, all coordinates are tried and the one which results in the neighbor with the largest separation distance is selected. If more coordinates result in the same separation distance, the one with the lowest index is selected. The fourth neighborhood is again very similar to the second neighborhood. The difference is that the first point is randomly selected from all points, instead of only the critical points. Although simulated annealing algorithms have been used before to deal with this type of problem, this adapted neighborhood structure, which is based on critical points, and the use of a different objective function, turned out to work well.

1.3 Packing problems : literature review

The problem of optimally placing n non-overlapping *objects* belonging to $\mathcal{R}^d$ of equal or different size, within a *smallest container* is a classical mathematical problem and has been widely considered in the literature. Besides being interesting because of their complexity, the attractiveness of packing problems is also motivated by a very broad range of practical applications. Packing problems arise in many scientific and engineering fields including production and packing for the textile, apparel, naval, automobile, aerospace and food industries, in particular, to problems related to Cutting and Packing (C&P) [56, 60, 61, 71, 97, 224, 229, 230, 253]. Packing problems are bottleneck problems in Computer Aided Design (CAD) and Computer Aided Manufacturing (CAM) where designs plans are to be generated for industrial plants, electronic modules, nuclear and thermal plants, etc. [224, 106]. In the pulp industry, packing of cylinders of pulp with different diameters and equal lengths into a shipping container is a common problem which is discussed in [75]. In [152] Liu and Chen discuss about the layout of newspapers or homepages. Dowsland et al. in [58] discuss a circle cutting problems arising in the motor cycle industry (manufacture of sprockets-toothed wheel used in a chain drive). The visualization of hierarchical information structures is an important topic in the visualization community [99]. In [250] Wang et al. applied the circle packing algorithms for visualization of large hierarchical data.

Although also higher dimensional packing problems have been considered (see, e.g., [62, 73, 74, 85, 98]), the literature about packing problems is mostly devoted to two-dimensional problems. Different two-dimensional objects have been considered. Chen et al. [39] investigated the mix-integer nonlinear model for constructing a 2-D selection problem, where placement objects were rectangles. The works of Marques et al. [166], Oliveira and Ferreira [189] and Correia et al. [42] discussed experiments with simulated annealing based meta-heuristic techniques for problems of allocating non-convex objects on rectangular containers. However, in the field of packing problems with two-dimensional objects, those dealing with circles (often equal ones) certainly play a major role (the mathematician R.L. Graham says that "the optimal packing of equal disks . . . is an ancient and extremely difficult problem. Some of these very simple problems, like how you pack 27 disks in a triangle, square, or circle, are very stubborn". For this reason a good deal of literature refers to the problem of packing circles into containers with different regular shapes (like squares, rectangles, triangles, circles). Probably, the most widely studied case is the problem of packing equal circles in a square. For this reason, although we will not focus our attention on this problem in the thesis, we will dedicate the next subsection to this special case, which has often inspired methods also for the other cases. Later on, we will dedicate a subsection to the problem of packing equal and unequal circles into a circular container, which will be the one considered in the thesis. We will not discuss packing in other shapes like triangles (see, e.g., [87, 171, 173, 185, 264]) or rectangular strips (see, e.g., [105, 139, 223, 225]). We also refer to the recent paper [35] for a detailed survey about methods and applications of circle packing problems.

1.3.1 Packing equal circles in a square

This problem is about 50 years old. In 1960, Moser was the first who studied circle packing in a square [182]. He guessed the optimal arrangement of 8 circles. Schaer and Meir [211] proved his conjecture and Schaer also solved the problem for $n = 9$ [208]. For $n \geq 10$ only the optimal packing of $n = 14, 16, 25, 36$ have been proved by hand. Wengerodt published proofs for $n = 14, 16, 25$ [255, 256, 257], while Wengerodt and Kirchner published a proof for $n = 36$ [130] by using theoretical tools. However, there are gaps in both proofs for $n = 25$ and 36 according to the review MR1453444 in Mathematical Reviews [235].

To tackle larger numbers of circles, researchers turned to computer-aided methods. By using them, optimal packings have been derived up to $n = 30$ [164, 165, 187, 194]. In [155] optimality within precision 10^5 has been proven for n up to 35 and for $n = 38, 39$. Computer-aided optimality proofs turn out to be quite computationally demanding. It is interesting to observe that these proofs are usually based on subdivisions of the unit square into non-overlapping sub-rectangles, each of which is guaranteed to contain at most one point of an optimal solution, and on the subsequent analysis of all the possible combinations of n such sub-rectangles. As a consequence, the computational burden does not increase regularly with n but has a sudden increase each time there is a need to increase the number of sub-rectangles (and then also the number of possible combinations) in order to guarantee that each of them contains at most one point of an optimal solution.

The difficulty of proving optimality led to the development of heuristic approaches aiming at improving best known results without giving optimality proofs for them. This represents the second main branch of research in the field of packing problems. Good approximate packings (i.e., packings determined by computer aided numerical computations without a rigorous proof) are reported in the literature for n up to 100 [235] (results for larger n values are also reported in the Packomania web site [274], but only a few methods have been run over such larger instances). At the same time, some other related results (e.g., patterns, bounds and some properties of the optimal solutions) were published as well [88, 231, 232, 235]. Below we give a short description of some of the employed methods.

de Groot et al. [91] searched for packings with $n \leq 22$ circles employing the simplex and quasi-Newton BFGS algorithm.

One technique that has proved effective simulates the idealized movements of billiard balls inside a circular or square table. In [88, 160, 162, 163] an event-driven (billiard balls) simulation algorithm has been applied for solving packing problems of equal circles: given a number of points-tiny disks-randomly spread out over a circular or square area, the disks move around like billiard balls, colliding, rebounding, and changing speed. As the disks roam, their diameters gradually increase, so the disks have less and less space within which to move. Eventually, they get locked into some sort of packing. The procedure is applied hundreds of times for a given number of disks, started in random positions and at random velocities.

Nurmela and Östergård, in [186] applied an energy minimization technique for solving the packing of equal circles problems. The authors define an energy function

$$E = \sum_{i \neq j} \left(\frac{\lambda}{d_{ij}} \right)^m$$

where d_{ij} is the Euclidean distance between points i and j, λ is a scaling factor, and m is a positive integer. They adopt a multistart approach starting a local optimization from at least 50 randomly generated solutions. Noting that when the energy is minimized, the corresponding solutions converge to those of the packing problem as $m \to \infty$, for each initial point the authors first perform a local search with a small m value and once they have reached a local minimum they increase the value of m, repeating this scheme until m becomes very large. The energy should be minimized over the box $[0, 1]^{2n}$ but the authors transform the problem into an unconstrained one by an appropriate change of variable. In the same paper the authors also recognize some regular patterns of disks which are optimal or presumably optimal for small n values but become non-optimal for n large enough. The best known among such patterns is the square lattice packing of $n = k^2$ points which is optimal for k up to 6 but is not for $k = 7$. Graham and Lubachevsky [88] considered the patterns proposed in [186] and extended them with new ones. Their billiards simulation method allows them to identify threshold indices above which it is guaranteed that the identified regular patterns become non-optimal.

Boll et al. proposed a two-phase approach in [27]. The first phase is an approximation one. During this phase each point in turn is moved along appropriately chosen directions with a step-size which is exponentially decreased during the run. The second phase is a refining one where the result of the first phase is the starting point for the billiards simulation method.

In [34] initially the unit square is subdivided into $k \times k$ sub-squares, where $k = \lceil \sqrt{n} \rceil$, and the initial solution is obtained by placing the n points at the center of n randomly selected distinct sub-squares. Then, each point is randomly perturbed and the perturbed point may be accepted even when it is non-improving (i.e., backtracking is allowed during the search). Starting from the results in [34], Szabó in [231] discussed some new regular patterns of points.

A recent development has been the application by Addis et al. [4] of Monotonic Basin Hopping and Population Basin Hopping approaches. This allowed to get many improvements of the best known solutions, even for n up to 100, now reported in [274].

For a more detailed history of the problem we refer to the recent book [234].

1.3.2 *Packing circles in a circle*

To the author's knowledge, the first reference to this problem dates back to Kravitz [136], where solutions for the problem of packing n identical circles in a minimal circular container are reported for n up to 19 without any optimality

proof. Reis [200] extended the range of n to 25.

Graham [86] proved optimality of packing with up to 7 circles. Fodor in [69, 70], exhibited the densest packing of $n = 12$ as well as $n = 19$ congruent circles in a circle with the help of a mathematical tool based on Besicovitch's lemma, developed by Bateman and Erdös [14].

Lubachevsky and Graham in [161] proposed a mathematical formulation for packing higher order identical circles in a large circle called curved hexagonal packing, when the number of circles can be formulated in a specific form. For 37, 61, and 91 disks, the curved hexagonal packings were the densest they obtained by computer experiments using the so-called 'billiards'simulation algorithm.

Huang and Xu [258] gave a quasi-physical personification algorithm based on combining the quasi-physical approach with the personification strategy by simulating the movement system for packing unequal and equal circles into a circle container.

An improved quasi-physical quasi-human (QPQH) algorithm has been given in [249]. This algorithm combines the quasi-physical approach and the quasi-human strategy.

The equivalent maximin distance problem for n points in a unit circle has been discussed and tackled with a standard greedy approach in [5].

In [268] Zhang and Huang presented a heuristic simulated annealing (HSA) algorithm to solve the (equal/unequal) circles packing in a circular container problem. For constructing a special neighborhood and jumping out of the local minimum trap, some effective heuristic strategies are incorporated in their SA based algorithm. The HSA algorithm inherits the merit of the SA algorithm, and can avoid the disadvantage of blind search in the simulated annealing algorithm to some extent according to the special neighborhood.

Zhang and Deng [269] proposed an hybrid algorithm for the packing of identical circles as well as unequal circles in a large circle. They combined the simulated annealing (SA) approach with tabu search (TS) approach to develop a hybrid algorithm to overcome the disadvantages of the two approaches taken by their own. The key of this algorithm lies in a powerful means for getting out of local minima. SA was introduced to escape from local optima with probability mechanism. TS is mainly used for preventing cycling and enhancing diversification. The computational results based on some benchmark instances showed that the hybrid algorithm was effective and robust, and almost always outperformed TS, SA and QPQH for all benchmark instances.

Mladenović et al. in [174] proposed a Reformulation Descent (RD) heuristic method, which iterates among several formulations of the same problem until local searches obtain no further improvement to pack equal circles into a unit circle. RD exploits the fact that a point which is stationary w.r.t. one formulation is not necessarily so with another. Therefore RD alternates between several formulations using a fast NLP code that stops in a stationary point.

PrunedEnriched-Rosenbluth Method (PERM) [90], also called population control algorithm, is a powerful strategy for pruning and enriching branches when searching the solution space and it has shown to be very efficient for solving protein folding problems [115] and [111]. In [159] Lü and Huang presented a new method that incorporates the PERM scheme into the strategy of maximum cave degree for (equal/unequal) circles packing in a circle. The basic idea of their approach is to evaluate the benefit of a partial configuration (where some circles have been packed and others are outside) using the principle of maximum cave degree, and use the PERM strategy to prune and enrich branches efficiently.

Huang et al. in [114] proposed two new heuristics to pack unequal circles into a two-dimensional circular container. In the first proposed heuristic they used the concept of `maximal hole degree` for selecting the next circle to place. In the second one they incorporate the concept of `self look-ahead strategy` to improve the first one. Recently, in [113] and [114] Huang et al. proposed a heuristic, based on the principle of maximum cave degree for corner-occupying actions (COAs), to select and pack the circles one by one, and they proposed a two level search strategy to improve the basic heuristic algorithm.

In [106] Hifi and M'Hallah proposeed a three-phase approximate algorithm. During its first phase, the algorithm successively packs the ordered set of circles. It searches for each circle its "best" position, given the positions of the already packed circles, where the best position minimizes the radius of the current containing circle. During its second phase, the algorithm tries to reduce the radius of the containing circle by applying (i) an intensified search, based on a reduction search interval, and (ii) a diversified search, based on the application of a number of layout techniques. Finally, during its third phase, the algorithm introduces a restarting procedure that explores the neighborhood of the current solution in search for a better ordering of the circles.

In [3] Addis et al. proposed a heuristic approach for the problem of placing n circles with increasing radii from 1 to n, which allowed them to win Al Zimmermann's Programming Contest about this problem. Their heuristic is based on the Monotonic and Population Basin Hopping approaches, but exploits the mixed nature, continuous (circle centers) and combinatorial (radii's values), of the problem to define proper perturbation moves. Moreover, some tricks are employed taking into account the special structure of the problem.

We finally remark that, as for the problem of packing equal circles in a square, also for the problem of packing equal circles in a circle best known results are reported and continuously updated in the Packomania web site [274].

1.4 Goals and outlook of the thesis

The main goals of the thesis are the following.

- We want to propose heuristic approaches for the maximin and packing problems considered in this thesis. In particular, we will consider Iterated Local Search (ILS) heuristics for the maximin LHD problem, and

Monotonic and Population Basin Hopping approaches for the problem of packing equal and unequal circles in a circular container.

- For each proposed approach we want to perform a careful analysis aimed at selecting in the most appropriate ways the main components and parameters on which the approach depends.
- We want to compare the proposed methods with those in the existing literature, in order to show that the proposed ones are competitive with (and, in some case, outperform) the already existing ones.

The thesis is organized as follows.

In **Chapter** 2 we present an overview about Iterated Local Search and Basin Hopping approaches. In order to show the superiority of MBH with respect to Multistart and the importance of the choice of the perturbation operator, an illustrating problem is also discussed.

Mathematical background of maximin LHD is briefly discussed in **Chapter** 3. We present the details of our proposed Iterated Local Search (ILS) approaches for finding maximin LHDs and we discuss two different objective functions driving the search within ILS. We discuss some local search procedures as well as perturbation moves.

Extensive experiments regarding several important issues about a first variant of ILS are performed in **Chapter** 4. In this chapter we compare experimentally such variant of ILS with a simple random search approach as well as with a Multistart approach. We present a comparison with some results available in the literature. We also propose an empirical formula related to the stopping rule of ILS. Finally, a complexity analysis mixing theoretical considerations and computational experiments is also carried one.

In **Chapter** 5 we discuss most of the issues already discussed in Chapter 4 but now related to a second variant of ILS.

In **Chapter** 6 we discuss the mathematical background about packing problems. We present the mathematical models of the packing problems which will be considered in the thesis.

In **Chapter** 7 Basing Hopping algorithms, namely Monotonic Basin Hopping (MBH) and Population Basin Hopping (PBH), are proposed for solving the problem of packing identical circles in a circular container with minimum radius. We propose different perturbation moves for the two approaches, and present dissimilarity measures, which are an important ingredient PBH approaches.

In **Chapter** 8 extensive experiments are performed regarding several issues related to MBH and PBH. Comparisons among different proposed versions of the algorithms are presented. Moreover we compare our experimental results with those available in the literature.

In **Chapter** 9 we propose algorithms for solving unequal circles packing prob-

lems. We propose some new ingredients which exploit the inequalities between circles' radii. We perform extensive experiments related to different issues of the proposed approaches, and make a comparison with the existing literature.

Finally in **Chapter** 10 we briefly discuss our motivation for considering two different (but, in fact, connected) types of problems as well as the corresponding solution approaches. We summarize our achievements and limitations of the proposed approaches and we also point out some possible future research directions.

Since both in maximin LHD and in packing problems we were able to improve some of the best known solutions, in **Appendix** A we list all such improvements for the two problems, while in **Appendix** B we report the coordinates or the figures of some of the improved solutions.

2. OVERVIEW OF HEURISTIC APPROACHES

When dealing with an optimization problem, the first aim is usually that of finding an optimal solution for it. Unfortunately, the intrinsic difficulty of the problem and/or the limited availability of computation time for the particular application from which the problem arises (think, e.g., about real-time applications, where solutions are required in very short times) may make computationally infeasible to return an optimal solution by the required time.

When solving to optimality of a problem is not possible, the only possible alternative is the use of *meta-heuristic* approaches. In particular, these approaches are usually of primary importance when dealing with problems with, among others, the following characteristics:

- NP-Hard;
- multi-modality (many local optima);
- non-differentiability or discontinuities (for continuous problems);
- good quality solutions, though not necessarily optimal, are searched for.

If the problem does not fit these requirements, one should probably search for other optimization tools, meta-heuristics should not be the best choice. Such approaches to optimization problems have developed dramatically in the last three decades. They have been successful in tackling many difficult problems, for which finding a solution in a straightforward manner is computationally infeasible, and have become more and more competitive. When designing a meta-heuristic, it is preferable for it to be simple, both conceptually and in practice. Naturally, it also must be effective, and if possible, general purpose. Of course, meta-heuristics offer no guarantee of obtaining the global solutions: ease of implementation and quickness have to be paid with the fact that even iterating might not provide a good enough solution for some instances. Although being general purpose is one of the requirements which should be fulfilled by a meta-heuristic, the quest for greater performance often suggests to incorporate problem-specific knowledge to increase efficiency, with the consequence of loosing both simplicity and generality [158].

The meta-heuristic approaches can be classified according to the particular characteristics of each algorithm. This classification leads to a better understanding of what strengths and shortcomings each method contains. Some of the most widely used meta-heuristic techniques are inspired from naturally occurring systems. The systems are based on biological evolution, intelligent problem solving, physical sciences and swarm intelligence, etc. Meta-heuristics can be classified into two broad classes: population-based methods and point-to-point methods.

In the latter methods, the search invokes only one solution at the end of each iteration from which the search will start in the next iteration. They can also be viewed as single-path search methods, where a single trajectory of solutions is followed during a run. On the other hand, the population-based methods invoke a set of many solutions at the end of each iteration. They can also be viewed as multi-path search methods, where different trajectories of solutions are followed in parallel during a run, and usually collaboration mechanisms exist which guarantee a sufficient diversification of the followed trajectories. Genetic algorithms [108, 83], Population Basin Hopping (PBH)[95] are examples of population-based methods; Simulated Annealing[131], Tabu Search [78, 79, 82], Iterated Local Search(ILS) [20, 19, 21], Monotonic Basin Hopping (MBH)[141] are examples of point-to-point methods.

In this chapter ILS and MBH meta-heuristic approaches are briefly presented and discussed. ILS approaches have been mainly applied to hard combinatorial optimization problems, where the search space is *discrete*. MBH approaches have been first applied to molecular conformation problems, which are special global optimization problems, and later extended to other *continuous* optimization problems. In fact, in spite of the different application fields, the description of the two methods will reveal that MBH can actually be viewed as a special ILS heuristic. However, in order to respect the existing literature and the different terminology sometimes employed in the two approaches, we will treat them separately. Besides ILS and MBH, we will also briefly discuss PBH, which can be viewed as a population based version of MBH.

2.1 ILS approach

Iterated Local Search(ILS) is a meta-heuristic designed to embed another, problem specific, local search as if it were a *black box*.This allows ILS to keep a more general structure than other meta-heuristics currently in practice. This simple type of search has been reinvented numerous times in the literature, with one of its earliest incarnations appearing in [154]. This simple idea [21] has a long history, and its rediscovery by many authors has lead to many different names for iterated local search like iterated descent [20, 19], large-step Markov chains [167], iterated Lin-Kernighan [123], chained local optimization [168], or combinations of these [7].

ILS has many of the desirable features of a meta-heuristic: it is simple, easy to implement, robust, and highly effective. The essence of the iterated local search meta-heuristic can be given in a nut-shell: one *iteratively* builds a sequence of solutions generated by the embedded heuristic, leading to far better solutions than if one were to use repeated random trials of that heuristic. Two main points in ILS are the following: (i) there must be a single chain that is being followed (this then excludes population-based algorithms); (ii) the search for better solutions occurs in a reduced space defined by the output of a black box heuristic. In practice, local search has been the most frequently used embedded heuristic, but in fact any optimizer can be used, be it deterministic or not. The essential idea of ILS lies in focusing the search not on the full space of solutions but on a smaller subspace defined by the solutions that are locally

optimal for a given optimization engine. The success of ILS lies in the biased sampling of this set of local optima. How effective this approach turns out to be depends mainly on the choice of the local search, of the perturbations, and of the acceptance criterion. So far, in spite of its conceptual simplicity, ILS can often become a competitive or even state of the art algorithm without the use of too much problem-specific knowledge. Perhaps this is because ILS is very malleable, many implementation choices being left to the developer [158]. In what follows we will give a formal description of ILS and comment on its main components.

2.1.1 Components of ILS Methods

The pseudo-code of ILS is the following

Procedure *Iterated Local Search*

s_0 = `GenerateInitialSolution`
s^* = `LocalSearch`(s_0)
repeat
s' = `Perturbation`(s^*, *history*)
$s^{*\prime}$ = `LocalSearch`(s')
s^* = `AcceptanceCriterion`($s^*, s^{*\prime}$, *history*)
until termination condition met
end

ILS involves four main components:

1. creating an initial solution;
2. a black-box heuristic that acts as a local search on the set S;
3. the perturbation operator, which modifies a local solution;
4. the acceptance criterion, which determines whether or not a perturbed solution will become the starting point of the next iteration.

In practice, much of the potential complexity of ILS is hidden in the history dependence. If there happens to be no such dependence, the walk has no memory: the perturbation and acceptance criterion do not depend on any of the solutions visited previously during the walk, and one accepts or not $s^{*\prime}$ with a fixed rule. This leads to random walk dynamics on S^* (where S^* is the locally optimal proper subset of S) that are "Markovian", the probability of making a particular step from s_1^* to s_2^* depending only on s_1^* and s_2^*. Most of the work using ILS has been of this type, though recent studies show unambiguously that incorporating memory enhances performance [226].

When exploring the search space, it is important for the ILS procedure to adequately search local regions. It is also important for the ILS procedure not to spend too much of its computational efforts around local optima, effectively limiting the search to a few regions of the domain. The former need describes the idea of *intensification* (ensuring that the process thoroughly inspects each local minimum) of the search, while the latter describes the idea of *diversification* (making sure the process is not searching a subset of the domain). Actually, it

is well known that the effective balancing of intensification and diversification is one of the largest hurdles encountered when tailoring any meta-heuristic (and not merely ILS) for a specific problem. Multiple strategies exist for accommodating both. Note that among the diversification strategies a simple, but widely employed, one is restarting: when no progress is observed, the algorithm is restarted from a new randomly generated solution.

Local search applied to the initial solution s_0 gives the starting point s^* of the walk in the set S^*. Starting with a good s^* can be important if high-quality solutions are to be reached as fast as possible. The initial solution s_0 used in the ILS is typically detected in one of two ways: (i) a random solution is generated or (ii) a greedy construction heuristic is applied. A greedy initial solution s_0 has two main advantages over random starting solutions: (i) when combined with local search, greedy initial solutions often result in better quality solutions s^*; (ii) a local search from greedy solutions takes, on average, less improvement steps and therefore the local search requires less CPU time. It has been shown, however, that this is true only in the short-term. Longer running algorithms see no significant difference in solution quality based on the initial solution [124, 227].

Ideally, the local search that provides the backbone of the ILS method should always return a local optimum and it should detect it as efficiently as possible. Since the behavior and performance of the overall ILS algorithm is quite sensitive to the choice of the embedded heuristic-*LocalSearch*, and since this step is usually the most time consuming (it occurs at each iteration of the meta-heuristic), one should optimize this choice whenever possible. Since ILS is usually applied to problems defined over discrete domains, choosing a good local search algorithm usually amounts to choosing a good neighborhood structure. In practice, there may be many different neighborhood structures and, consequently, many different local search algorithms that can be used for the embedded heuristic. One might think that the better the quality of the solutions returned by the local search algorithm, the better the corresponding ILS [124, 228]. But if we assume that the total computation time is fixed, it might be better to apply more frequently a faster but less effective local search algorithm than a slower and more powerful one. Clearly which choice is best depends on just how much more time is needed to run the better heuristic. If the speed difference is not large, for instance if it is independent of the instance size, then it is usually worth using the better heuristic. In fact, it is difficult to make *a priori* this choice, as well as many others which have to be done when defining a ILS heuristic. A good strategy is always that of making choices only after extensive computational experiments.

The main drawback of any local search algorithm is that, by definition, it gets trapped in local optima that might be significantly worse than the global optimum. Some meta-heuristic approaches like Simulated Annealing or Tabu Search, try to overcome this limitation by the introduction of non-improving move: local searches follow descent trajectories which are unable to make any progress once a local minimum has been reached, while non-improving moves allow to escape from local minima through hill-climbing (or backtracking). The strategy employed by ILS to escape from local optima is represented by per-

turbations to the current local minimum. The perturbation scheme takes a locally optimal solution, $s*$, and produces another solution from which a local search is started at the next iteration. Hopefully, the perturbation will return a solution outside the basins of attraction of previously visited local minima. That is, it will be "near" a previously unvisited local optimum. Choosing the correct perturbation scheme is of primary importance, because it has a great influence on the intensification/diversification characteristics of the overall algorithm. Generally, the local search should not be able to undo the perturbation, otherwise one will fall back into the local optimum just visited. Perturbation schemes are commonly referred to as "strong" and "weak ", depending on how much they affect the solution that they change. A perturbation scheme that is too strong has too much diversity and will reduce the ILS to an iterated random restart heuristic. A perturbation scheme that is too weak has too little diversity and will result in the ILS not searching enough of the search space. The perturbation scheme should be chosen in such a way that it is as weak as possible while still maintaining the following condition: the likelihood of revisiting the perturbed solution on the next execution of `LocalSearch` should be low [158]. The strength should remain as low as possible to speed up execution time. The desired perturbation scheme will return a solution near a locally optimal value. If this is the case, the local search algorithm should take less time to reach the next locally optimal value. Components from other meta-heuristics can sometimes be incorporated into the perturbation phase. Battiti and Protasi [17] use memory structures similar to tabu search [78, 79, 82] to control the perturbation. In doing so, one can force intensification when globally *good* values are reached and force diversification when the search stagnates in an area of the search space. Borrowing from Simulated Annealing [131], temperature controlled techniques have been used to force the perturbation to change in a deterministic manner. Basic variable neighborhood search employs a deterministic perturbation scheme.

When the current solution (which is a local optimum), s^*, is perturbed, the result is the new solution s'. s' is then passed to the black-box search heuristic i.e. `LocalSearch`. The resulting local optimum $s^{*\prime}$ must pass acceptance criterion for $s^{*\prime}$ to be designated as the new "current solution". Just as perturbation can range from too much intensification (no perturbations) to too much diversification (perturb all elements of the solution), acceptance criterion choices affect the search in a similar way. The most dramatic acceptance criterion on the side of diversification is to accept all perturbed solutions. This type of practice can undermine the foundations of ILS, since it encourages a "random-walk" type search. Contrasting with this, the algorithm accept only solutions that are improvements to the globally optimal value (a sort of greedy strategy). Many implementations of ILS employ this type of acceptance strategy [202]. This type of criterion, especially with a weak perturbation scheme, can restrict the search from escaping the current basin of attraction. Moreover, with this type of scheme the probability of reaching the same locally optimal value increases a trait that reduces the algorithm's overall effectiveness. In this case random restart when the search stagnates is a good way to ensure some diversification and to counterbalance the (possible) negative effects of too greedy a search. Large perturbations are only useful if they can be accepted. This only occurs if the acceptance criterion is not too biased toward better solutions

[157]. The tabu search relies on occasionally moving the search into areas with worse objective functions in order to better search the solution space. In [226] author shows that acceptance criteria that accept some worse solutions outperform their best-only counterparts.

For what concerns the stopping rule, generally the algorithm executes until one of the following conditions is met:

- a predetermined number of cycles have occurred;
- the best solution has not changed for a predetermined number of cycles;
- a solution has been found that is beyond some predetermined threshold.

Notice that the three rules constitute three different approaches: the first rule executes independently of the performance of the process (*time*); the second one stops executing when the performance of the method stops improving (*performance*); the third one stops executing when a solution is found that is "good enough" (*utility*) .

The rationale behind ILS is supported by the proximate optimality principle [82]. This principle assumes that good solutions are similar. This assumption is reasonable for most real-world problems. For example, the percentage of common edges in any two locally optimal solutions obtained by the Lin-Kernighan method is about 85 % on average. Based on this principle, search should take place in S^* around the best locally optimal solutions found so far. Perturbations should be such that the structure of a local solution is not disrupted and many of its "good" parts are also retained by the perturbed solutions and by the local optimum reached when starting a local search from the perturbed solution. Even when using the most naïve implementations of the main components of ILS, one can do much better than with random restart. But with further work so that the different modules are well adapted to the problem at hand, ILS can often become a competitive or even state of the art algorithm. This dichotomy is important because the optimization of the algorithm can be done progressively, and so ILS can be kept at any desired level of simplicity. This, plus the modular nature of ILS, leads to short development times and gives ILS an edge over more complex meta-heuristics in the world of industrial applications. As an example of this, recall that ILS essentially treats the embedded heuristic as a black box; then upgrading an ILS to take advantage of a new and better local search algorithm is nearly immediate.

2.2 MBH approach

Monotonic Basin Hopping (MBH) is a heuristic approach for the global optimization of high-dimensional and highly multi-modal continuous functions. It has been first applied in the field of molecular conformation problems (see [141, 247]), where the global optimization of the mathematical model of the energy of a cluster of atoms allows to predict the geometrical structure of such cluster. MBH falls into the category of methods in which the function to be optimized is transformed to make searching easier without affecting the solution. In MBH the transformation maps the function onto a series of plateaus where

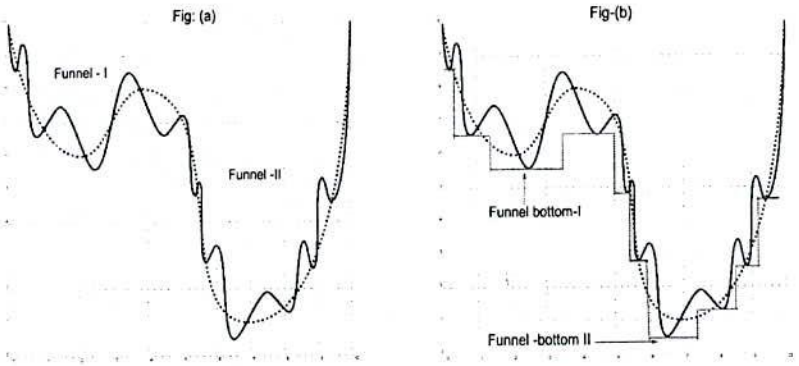

Fig. 2.1: A schematic diagram illustrating Funnel and MBH approach on one dimensional example

the barriers between local minima have been removed [141] (see Figure 2.1(b)).

The key idea of this approach is the measurement of the difficulty of the problems by the concept of *funnel* (see again Figure 2.1(b)). This concept was first introduced in the previously mentioned global optimization problems arising in computational chemistry. For many molecular conformation potential energy surfaces, the local minima can be organized by a simple adjacency relation into a single or at most a small number of funnels. A distinguished local minimum lies at the bottom of each funnel and a monotonically descending sequence of adjacent local minima connects every local minimum in the funnel with the funnel bottom. Thus the global minimum can be found among the comparatively small number of funnel bottoms, and a multistart strategy based on sampling funnel bottoms becomes viable.

In order to roughly describe what a funnel is, here we give an definition based on neighborhoods of local minima (see also [4]). Let $\mathcal{N}$ be a neighborhood structure defined upon the set $\mathcal{X}$ of all local minima of a given objective function f. Then, a funnel can be defined as a maximal subset $\mathcal{Y} \subseteq \mathcal{X}$ of local minima with the following property: there exists a local minimum $\overline{X} \in \mathcal{Y}$ such that for all $X \in \mathcal{Y}$ a decreasing sequence of neighbor local minima in $\mathcal{Y}$ starting at X and ending at $\overline{X}$ exists, i.e.

$$\begin{aligned} \exists\, X_0, X_1, \cdots, X_t : X_i \in \mathcal{N}(X_{i-1}) \cap \mathcal{Y} \qquad & i = 1, 2, \cdots, t \\ f(X_i) < f(X_{i-1}) \qquad & X_0 = X, \quad X_t = \overline{X}. \end{aligned}$$

The common final endpoint of the sequences is called funnel bottom. We can also think of a graph whose nodes are local optima; two local optima X_i and X_j with $f(X_i) \leq f(X_j)$ are connected by a directed arc if from X_i it is possible to reach X_j. This possibility might be interpreted and defined in different ways. In chemistry and biology reachability corresponds to the situation in

which there exists a continuous path connecting the two configurations which never exceeds a given energy level. So we might define as connected by an arc two local minima such that there is a path connecting them along which the objective function never exceeds a given value (the red path in Figure 2.1(b)). Alternatively, we might say that X_j is reachable from X_i if a local optimization started from a point in a neighbor of X_i ends up at X_j . In any case, given a definition of reachability, a funnel bottom is defined as a local minimum with no outgoing arcs and a funnel is defined as a maximal set of local optima from which the same funnel bottom can be reached through a directed path. Thus, a funnel is a set of local minima characterized by the fact that for each of them there exists at least one decreasing sequence of "neighbor " local minima along a path leading to a unique local minimum corresponding to the bottom of the funnel. The number of funnels, together with their width, seems to be a much more appropriate measure for characterizing difficult GO problems with respect to the overall number of local minima.

There exist in the literature simple but quite effective algorithms which are particularly well suited for functions of the above type: the Basin Hopping (BH) algorithm by Wales and Doye [247] and, the Monotonic Basin Hopping (MBH) algorithm by Leary[141] and some of its variants [2] proved to be extremely efficient in detecting funnel bottoms. The basic structure of MBH, as given in [141] is the following, where `MaxNoImp` is a prefixed parameter.

MBH: let X: initial local minimum
Step 1. Compute $Y := \Phi(X)$ such that $Y \in \mathcal{N}(X)$
Step 2. if $f(Y) < f(X)$ **then** set $X := Y$;
else reject Y ;
Step 3. Repeat Steps 1- 2 until
`MaxNoImp` consecutive rejections have occurred;
Step 4. **return** X;

The local move Φ is usually defined as

$$\Phi(X) = L_f(X + \Delta),$$

where Δ is usually a uniform random vector drawn from a box with given size. We observe that MBH performs a kind of monotonic depth-first search in search space S. Despite its simplicity, computational experiments reveal the effectiveness of MBH when faced with GO problems with single funnel landscapes or with a large basin of attraction of the funnel containing the global optimum [2, 141]. In fact, MBH cleverly copes with the structure of a funnel, generating a descent sequence of local minima; the current best solution is (heuristically) declared to be a funnel bottom after `MaxNoImp` non-improving iterations.

But if we have a closer look to MBH, it will become immediately clear what we stated in the introduction of this chapter, i.e. that MBH is in fact nothing but an ILS heuristic. Indeed, Φ is nothing but the perturbation operator, the acceptance criterion is the monotonic one (only accepts improving moves), and the stopping criterion asks for stopping when no improvement is observed for a given number (`MaxNoImp`) of iterations.

2.3 Illustrating MBH (and ILS) through a toy problem

In this section we present a simplified toy problem through which we show both the superiority of MBH with respect to a Multistart strategy (multiple local searches started from randomly generated points), and the importance of the choice of the perturbation operator (in particular, in this case we will only consider the dependency on the size of the perturbation). These issues will also be more thoroughly computationally investigated for the problems considered in this thesis, but the theoretical analysis of this simple example will also be useful. We only discuss MBH but a completely similar analysis could also be performed for any ILS approach. We will assume that our problem is a two-dimensional one with a set $\mathcal{S}^*$ made up by 25 local minimizers defined as follows

$$\mathcal{S}^* = \{(i,j) \ : \ i,j \in \{-2,-1,0,1,2\}\}.$$

The objective function is simply x^2+y^2, so that the global minimizer is obviously the origin. We build a graph G whose nodes are the local minimizers and two nodes/local minimizers are connected by an edge if they have a common coordinate value and a difference of one for the other coordinate value. We will make the following simplifying assumptions:

- the computational effort to detect a local minimizer is the same for all the 25 local minimizers;
- the size of the region of attraction of each local minimizer is the same, or, equivalently, all local minimizers have the same probability 1/25 of being detected when starting a local search from a point uniformly generated over the feasible region.

The first assumption allows us to evaluate the effort to detect the global minimizer in terms of number of local searches performed. The second assumption immediately tells us that

$$ELC[Multi] = 25,$$

i.e., the expected number of local searches (ELC) for Multistart is equal to 25.

Now, let us compute the same value ELC for MBH. We first consider the case in which the neighborhood of a local minimizer employed for the perturbation is the one made up by all the local minimizers which can be reached from the current one through a path of length one in graph G (more simply, those local minimizers connected by an edge to the current one). This neighborhood will be denoted by $\mathcal{N}_1$. Because of symmetry, we can group together some local minimizers. In particular, we will recognize six different groups:

$S_1 = \{(-2,-2);\ (-2,2);\ (2,-2);\ (2,2)\}$
$S_2 = \{(-2,-1);\ (-2,1);\ (2,-1);\ (2,1);\ (-1,-2);\ (1,-2);\ (-1,2);\ (1,2)\}$
$S_3 = \{(0,2);\ (2,0);\ (-2,0);\ (0,-2)\}$
$S_4 = \{(-1,-1);\ (-1,1);\ (1,-1);\ (1,1)\}$
$S_5 = \{(1,0);\ (0,1);(-1,0);\ (0,-1)\}$
$S_6 = \{(0,0)\}$

We can represent the evolution of MBH through a Markov chain whose states are the six groups above, and whose matrix of transition probabilities (assuming

a uniform distribution over the neighborhood of a given local minimizer) is the following

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ 0 & 0 & \frac{2}{3} & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The expected time to reach the global minimizer is equal to the expected time to reach the unique absorbing state (which, indeed, is the global minimizer). In order to compute the expected absorbing time from each starting state, we need to solve the following linear system

$$\begin{cases} \mu_1 = \mu_2 + 1 \\ \mu_2 = \frac{1}{3}\mu_2 + \frac{1}{3}\mu_3 + \frac{1}{3}\mu_4 + 1 \\ \mu_3 = \frac{2}{3}\mu_3 + \frac{1}{3}\mu_5 + 1 \\ \mu_4 = \frac{1}{2}\mu_4 + \frac{1}{2}\mu_5 + 1 \\ \mu_5 = \frac{1}{2}\mu_5 + \frac{1}{2}\mu_6 + 1 \\ \mu_6 = 0 \end{cases}$$

The solution of this system is the following

$$\mu_1 = 7 \quad \mu_2 = 6 \quad \mu_3 = 5 \quad \mu_4 = 4 \quad \mu_5 = 2 \quad \mu_6 = 0$$

Taking into account that the initial distribution of all local minimizers is the uniform one, we have that

$$ELC[MBH(\mathcal{N}_1)] = \frac{4}{25}\mu_1 + \frac{8}{25}\mu_2 + \frac{4}{25}\mu_3 + \frac{4}{25}\mu_4 + \frac{4}{25}\mu_5 + 1 = \frac{29}{5},$$

which is much inferior with respect to $ELC[Multi]$.

Next we try to show the dependency of the results form the size of the perturbation operator. We define a new neighborhood, denoted by $\mathcal{N}_2$, made up by all local minimizers reachable with a path of length at most 2 in graph G starting from the current local minimizer. Note that $\mathcal{N}_2$ is obviously a larger neighborhood with respect to $\mathcal{N}_1$. Again we can perform an analysis based on Markov chains. We group the local minimizers as above. The matrix of transition probabilities is now the following

$$\begin{bmatrix} 0 & \frac{2}{5} & \frac{2}{5} & \frac{1}{5} & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & \frac{1}{8} \\ 0 & 0 & 0 & \frac{7}{10} & \frac{1}{5} & \frac{1}{10} \\ 0 & 0 & 0 & 0 & \frac{10}{11} & \frac{1}{11} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

In order to compute the expected absorbing time from each starting state, we need to solve the following linear system

$$\begin{cases} \mu_1 = \frac{2}{5}\mu_2 + \frac{2}{5}\mu_3 + \frac{1}{5}\mu_4 + 1 \\ \mu_2 = \frac{1}{2}\mu_2 + \frac{1}{6}\mu_3 + \frac{1}{6}\mu_4 + \frac{1}{6}\mu_5 + 1 \\ \mu_3 = \frac{1}{2}\mu_3 + \frac{1}{4}\mu_4 + \frac{1}{4}\mu_5 + \frac{1}{8}\mu_6 + 1 \\ \mu_4 = \frac{7}{10}\mu_4 + \frac{1}{5}\mu_5 + \frac{1}{10}\mu_6 + 1 \\ \mu_5 = \frac{10}{11}\mu_5 + \frac{1}{11}\mu_6 + 1 \\ \mu_6 = 0 \end{cases}$$

The solution of this system is the following

$$\mu_1 = \frac{41}{3} \quad \mu_2 = \frac{27}{2} \quad \mu_3 = \frac{77}{6} \quad \mu_4 = \frac{32}{3} \quad \mu_5 = 11 \quad \mu_6 = 0$$

Taking into account that the initial distribution of all local minimizers is the uniform one, we have that

$$ELC[MBH(\mathcal{N}_2)] = \frac{4}{25}\mu_1 + \frac{8}{25}\mu_2 + \frac{4}{25}\mu_3 + \frac{4}{25}\mu_4 + \frac{4}{25}\mu_5 + 1 = \frac{977}{75},$$

which is inferior with respect to $ELC[Multi]$ but clearly superior to $ELC[MBH(\mathcal{N}_1)]$, thus showing that a smaller neighborhood is more appropriate in this case.

The assumption that the distribution of all local minimizers is the uniform one is essential for the above result. If we drop it we can get to different conclusions. In particular, if we assume that all local minimizers, except the global one, have a small probability $\varepsilon > 0$ but the global one has a very large probability $1 - 24\varepsilon$, then we might expect that Multistart works better than MBH($\mathcal{N}_1$). This can also be confirmed through the usual analysis. It holds that

$$ELC[Multi] = \frac{1}{1-24\varepsilon} = 1 + \frac{24\varepsilon}{1-24\varepsilon}.$$

The matrix of transition probabilities (assuming that each transition is the restriction of the general distribution of local minimizers over the neighborhood $\mathcal{N}_1$ of a given local minimizer) is the following

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ 0 & 0 & \frac{2}{3} & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & \frac{3\varepsilon}{1-21\varepsilon} & \frac{1-24\varepsilon}{1-21\varepsilon} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

In order to compute the expected absorbing time from each starting state, we need to solve the following linear system

$$\begin{cases} \mu_1 = \mu_2 + 1 \\ \mu_2 = \frac{1}{3}\mu_2 + \frac{1}{3}\mu_3 + \frac{1}{3}\mu_4 + 1 \\ \mu_3 = \frac{2}{3}\mu_3 + \frac{1}{3}\mu_5 + 1 \\ \mu_4 = \frac{1}{2}\mu_4 + \frac{1}{2}\mu_5 + 1 \\ \mu_5 = \frac{3\varepsilon}{1-21\varepsilon}\mu_5 + \frac{1-24\varepsilon}{1-21\varepsilon}\mu_6 + 1 \\ \mu_6 = 0 \end{cases}$$

Taking into account that the initial distribution of all local minimizers , we have after some computations that

$$ELC[MBH(\mathcal{N}_1)] = 1 + 88\varepsilon + o(\varepsilon),$$

which, for ε small enough, is larger than $ELC[Multi]$.

2.4 Population Basin Hopping

The solution of GO problems with a large number of funnels and/or a small basin of attraction for the funnel containing the global minimum is a much more difficult task. In these cases MBH often fails to reach the global optimum. In these cases many runs of MBH might be needed before ending up in the funnel bottom corresponding to the global optimum, i.e. many different single paths have to be followed. Basically, diversification is ensured through multiple restarts from randomly generated solutions. Unfortunately, a possible drawback is that many runs end up in a strongly attractive funnel bottom which, however, does not correspond to the global optimum.

In these cases population based approaches might be better suited to solve such difficult problems. These approaches rely on an evolving collection – population – of solutions; evolution is driven by perturbation operators (crossover, mutation) and selection/replacement mechanisms. Such mechanisms embed some device to enforce diversity among members of the population, in order to avoid that all or most of them converge to the same solutions.

Inspired by the Conformational Space Annealing algorithm (see, e.g., [144]) in which the single path search is substituted by a multiple path search, Grosso et al. [95] proposed population based MBH approach called Population Basin Hopping(PBH) approach. Rather than following a single search like in MBH, PBH searches over the solutions in a multi-search fashion. During the search, members of the population collaborate with each other in order to guarantee *diversification* of the search and to avoid the *greediness* which might characterize a single path search. This way the previously mentioned drawback of multiple runs of MBH ending up in the same non-global funnel bottom, is avoided by keeping far from each other the different paths followed during the search. When dealing with particularly hard molecular conformation problems (Morse clusters with large ρ values), PBH turned out to be quite clearly more efficient than MBH [95].

Including all components of MBH approach, there exist a new operator in PBH approach called *dissimilarity measure* $\mathcal{D}$. The operator *dissimilarity measure* measures the diversity between two solutions. The concept of similarity is problem-specific; the only essential requirement for the similarity of two solutions, say, X, Y is $\mathcal{D}(X, Y) \equiv 0$ if $X = Y$ [95]. The dissimilarity measure is employed to avoid that "too similar" solutions are present at the same time within the current population. In particular, two equivalent solutions can not be present at the same time within the population. Ideally, this should bring to the situation where all the paths followed during a run of PBH end up in a distinct funnel bottom, thus avoiding the waste of computational effort caused

by multiple detection of the same funnel bottom.

We do not give here the details of PBH and postpone their presentation to Section 7.2, where we will present a PBH approach for packing problems. For further details, we also refer to the paper [95].

3. MAXIMIN LHD

We will denote as follows the s-norm distance between two points x_i and x_j, $\forall \quad i,j = 1,2,\cdots,N$:

$$d_{ij} = \|x_i - x_j\|_s , \tag{3.1}$$

Unless otherwise mentioned, we will only consider the Euclidean distance measure ($s = 2$). In fact, we will usually consider the squared value of d_{ij} (in brief d), i.e. d^2 (saving the computation of the square root). This has a noticeable effect on the execution speed since the distances d will be evaluated many times.

3.1 Definition of LHD

A Latin Hypercube Design (LHD) is a statistical design of experiments, which was first defined in 1979 [169]. An LHD of k-factors (dimensions) with N design points, $x_i = (x_{i1}, x_{i2} \cdots x_{ik}) : i = 0,1,\ldots,N-1$, is given by a $N \times k$- matrix (i.e. a matrix with N rows and k columns) X, where each column of X consists of a permutation of integers $0,1,\cdots,N-1$ (note that each factor range is normalized to the interval $[0, N-1]$) so that for each dimension j all $x_{ij}, i = 0,1,\cdots,N-1$ are distinct. We will refer to each row of X as a (discrete) design point and each column of X as a factor (parameter) of the design points.

We can represent X as follows

$$X = \begin{pmatrix} x_0 \\ \vdots \\ x_{N-1} \end{pmatrix} = \begin{pmatrix} x_{01} & \cdots & x_{0k} \\ \vdots & \ldots & \vdots \\ x_{(N-1)1} & \cdots & x_{(N-1)k} \end{pmatrix} \tag{3.2}$$

such that for each $j \in \{1,2\cdots,k\}$ and for all $p,q \in \{0,1,\cdots,N-1\}$ with $p \neq q$ $x_{pj} \neq x_{qj}$ holds.

Given a LHD X and a distance d, let

$$D = \{d(x_i,x_j)\colon 1 \leq i < j \leq N\}.$$

Note that $|D| \leq \binom{n}{2}$. We define $D_r(X)$ as the r-th minimum distance in D, and $J_r(X)$ as the number of pairs $\{x_i,x_j\}$ having $d(x_i,x_j) = D_r(X)$ in X.

The maximin LHD problem aims at finding a LHD X^* such that $D_1(X)$ is as large as possible. However, a search which only takes into account the D_1 values is certainly not efficient. Indeed, the landscape defined by the D_1 values is "too flat". For this reason the search should be driven by other optimality criteria, which take into account also other values besides D_1. In what follows we will present some of them.

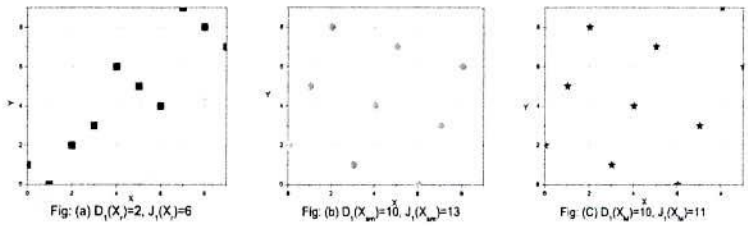

Fig. 3.1: Some LHDs and their corresponding (D_1, J_1) values.

3.1.1 Optimality Criteria

In order to drive the search through LHDs we need some criterion to compare them. Below we will describe some of the criteria employed in the literature.

Opt(D_1, J_1) Optimality Criterion : Under this criterion a LHD Y can be considered better than another one X if a lexicographic ordering holds:

$$\begin{array}{lll} D_1(Y) > D_1(X) & \text{or} & \\ D_1(Y) = D_1(X) & \text{and} & J_1(Y) < J_1(X). \end{array} \tag{3.3}$$

In what follows we illustrate this optimality criterion through some examples. In Figure 3.1(a) X_r is a randomly generated LHD with $(k, N) = (2, 10)$ where $D_1(X_r) = 2$ and $J(X_r) = 6$; Figure 3.1 (b) presents an improved configuration X_{sm} where $D_1(X_{sm}) = 10$ with $J(X_{sm}) = 13$. A third LHD X_M is given in Figure 3.1 (c) where $D_1(X_M) = 10$ and $J_1(X_M) = 11$; by the Opt(D_1, J_1) criterion this is the best configuration among the three.

By generalizing this approach, we can consider the problem like a *multi-objective problem with priorities*: maximize the objective with highest priority D_1; within the set of optimal solutions with respect to D_1, minimize the objective with second highest priority J_1. Note that Johnson et. al. [125] first proposed this optimality criterion.

Opt(ϕ) Optimality Criterion : As previously remarked, if there exist different LHDs with equal D_1 and J_1 values, i.e. in case there exist at least two LHDs X, Y such that $D_1(X) = D_1(Y) = D_1$ and $J_1(X) = J_1(Y) = J_1$, we could further consider the objective D_2 and maximize $D_2(X)$, the second smallest distance in X, and, if equality still holds, minimize $J_2(X)$, the number of occurrence of $D_2(X)$, and so on. Then an optimal design X sequentially maximizes D_is and minimizes J_is in the following order: $D_1, J_1; D_2, J_2, \cdots, D_m, J_m$. Morris and Mitchell [178] have used all the above measures to define a family of scalar-valued functions (to be minimized), which can be used to rank competing designs in such a way that a maximin design receives the highest ranking. This family of functions, indexed by p, is given by

$$\phi_p(X) = \sum_{r=1}^{m} \left[\frac{J_r(X)}{(D_r(X))^p} \right]^{\frac{1}{p}}, \tag{3.4}$$

where p is a positive integer parameter. Under this criterion, LHD Y is better than X if

$$\phi_p(Y) < \phi_p(X).$$

Note that for large enough p, each term in the sum in (3.4) dominates all subsequent terms. Through p we can control the impact of the different D_r distances: as p increases, the impact of distance D_1 becomes more and more relevant. In the form (3.4), the evaluation of ϕ_p would be computationally costly. However, it has a computationally cheaper form (see [122]). Indeed, (3.4) can be simplified as

$$\phi_p(X) = \left[\sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{1}{d_{ij}^p} \right]^{\frac{1}{p}}, \tag{3.5}$$

which can be computed without the need of detecting and ordering all the D_i values.

An apparent drawback of the Opt(ϕ) criterion, if we are interested in maximin values (maximum D_1 value), is that LHDs with smaller (better)ϕ_p can have a worse(smaller) D_1, i.e. we can have X and Y such that $\phi_p(X) < \phi_p(Y)$ and $D_1(X) < D_1(Y)$. This phenomenon has been frequently observed in our computational experiments (see Section 5.1). Nevertheless, a profitable choice is to work in order to minimize the ϕ_p function but, at the same time, keep track of the best (D_1, J_1) values observed during such minimization. This way the search in the solution space is guided by a kind of heuristic function. Such a mixed approach might appear strange but, as we will demonstrate experimentally, it can be extremely effective.

While the two criteria above are strictly related to maximin values and, as we will see, they will be widely employed in the definition of approaches for detecting maximin solutions, for the sake of completeness, we also mention that also other optimality criteria, not necessarily related with maximin values, are available in the literature. Below we present a couple of them. We point out that, while we could specialize our approach to these criteria, in the thesis we have not pursued experiments with them.

Force Optimality Criterion : At first we introduce a function G which, in a physical analogy, is the sum of the magnitudes of repulsive forces if the sample points are considered as electrically charged particles (see [15, 151]).

$$G(X) = \sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{1}{d_{ij}^2} \tag{3.6}$$

From the point of view of the physical analogy, it would have been natural with the power 1 (instead of 2) in the denominator of the terms of the sum. However, as we stated earlier, with the power 2 a computation of a square root for each term is avoided, so that we have a cheaper computation. Then Y is better than X if $G(Y) < G(X)$.

Correlation Optimality Criterion: Iman and Conover [119], Owen[192], and Tang [238] propose to choose designs by minimizing correlations among factors within the class of LHDs. Owen, in [192], used the following performance measure for evaluating the goodness of the LHD with respect to pairwise correlations. It is defined as follows

$$\rho^2 = \frac{\sum_{i=1}^{k} \sum_{j=i+1}^{k} \rho_{ij}^2}{k(k-1)/2} \tag{3.7}$$

For calculating the correlation between each single pair of factors (say column q and column r), here we use Pearson's formula:

$$\rho_{qr} = \frac{N\sum x_{iq}x_{ir} - \sum x_{iq}\sum x_{ir}}{\sqrt{N\sum x_{iq}^2 - (\sum x_{iq})^2}\sqrt{N\sum x_{ir}^2 - (\sum x_{ir})^2}}, \tag{3.8}$$

where the sums are over $i = 1, \cdots, N$. Here Y is better than X if $\rho^2(Y) < \rho^2(X)$.

3.2 Proposed ILS heuristic for maximin LHD

In Section 2.1 we have discussed a general scheme for ILS-based algorithms. Now we present the ILS based procedure for maximin Latin hypercube design. As we have stated earlier, the main components of ILS heuristic approaches are Initialization ($\mathcal{I}_{\mathcal{S}}$), LocalSearch ($\mathcal{L}_{\mathcal{M}}$), Perturbation Move ($\mathcal{P}_{\mathcal{M}}$), and the Stopping Rule ($\mathcal{S}_{\mathcal{R}}$); the pseudo-code of the proposed ILS heuristic for maximin LHD problems is the following:

Step 1. **Initialization :** $X = \mathcal{I}_{\mathcal{S}}(\{0, 1, \ldots, N-1\}))$
Step 2. **Local Search :** $X^* = \mathcal{L}_{\mathcal{M}}(X)$
while $\mathcal{S}_{\mathcal{R}}$ not satisfied **do**
  Step 3. **Perturbation Move** :$X' = \mathcal{P}_{\mathcal{M}}(X)$
  Step 4. **Local Search :** $X^* = \mathcal{L}_{\mathcal{M}}(X')$
  Step 5. **Improvement test :** if X^* is better than X,
  set $X = X^*$
end while
Return X

Below we detail the components in order to fully specify our algorithm.

3.2.1 Initialization ($\mathcal{I}_{\mathcal{S}}$)

The initialization ($\mathcal{I}_{\mathcal{S}}$) procedure embedded in our algorithm is extremely simple: the first initial solution is randomly generated. In particular, the first initial solution generation is built as follows. For each component $h \in \{1, \ldots, k\}$ a random permutation $v_0, \ldots, v_{N-1}$ of the integers $0, 1, \ldots, N-1$ is generated and we set

$$x_{rh} = v_r \quad \text{for all } r \in \{0, \ldots, N-1\}.$$

Although more aggressive procedures could be designed, we chose random generation because it is fast and unbiased.

3.2.2 Local Search Procedure ($\mathcal{L}_{\mathcal{S}}$)

In order to define a local search procedure ($\mathcal{L}_{\mathcal{S}}$), we need to define a concept of *neighborhood* of a solution. Given a LHD $X = (x_1, \ldots, x_N)$, its neighborhood is made of all other LHDs obtained by applying *local moves* to X. Before introducing some *local moves*, we first introduce the notion of *critical point*.
Critical point: We say that x_i is a critical point for X, if

$$\min_{j \neq i} d(x_i, x_j) = D_1(X),$$

i.e., the minimum distance from x_i is also the minimum one among all the distances in X. We denote by $\mathcal{I}(X) \subseteq \{1, \ldots, N\}$ the set of indices of the critical points in X.

Local moves($\mathcal{LM}$): A local move is an operator that applies some form of slight perturbation to a solution X, in order to obtain a different solution. Different local moves define different neighborhoods for local search. In the literature two different local moves are available: Rowwise-Pairwise (RP) exchange [193] and Columnwise-Pairwise (CP) exchange [178]. In Park's algorithm [193] some active pairs (pairs of critical points, in our terminology) are selected. Then, for each chosen pair of two active rows, say i_1 and i_2, the RP exchange algorithm considers all the possible exchanges of corresponding elements as follows:

$$x_{i_1,p} \leftrightarrow x_{i_2,q} \; \forall \; p, q = 1, 2, \ldots, k : p \neq q,$$

and finds the best exchange among them. The CP algorithm proposed by Morris and Mithchell [178] exchanges two randomly selected elements within a randomly chosen column. But in [150], Li and Wu defined the CP algorithm in a bit different way: they randomly choose a column and replace it by its random permutations if a better LHD is obtained.

Although we have also developed CP based local moves, in our proposed algorithms we mainly consider RP based local moves but a bit different than those of [193]. We propose two types of RP moves related to two different optimality criteria — namely Opt(D_1, J_1) and Opt(ϕ).

For Opt(D_1, J_1), we propose Rowwise-Pairwise Critical Local Moves (we call it $\mathcal{LM}_{RpD1}$) as follows. The algorithm sequentially chooses two points (rows) such that at least one of them is a critical point, then exchanges two corresponding elements (factors) of the selected pair. If $i \in \mathcal{I}(X)$, $r, j \in \{1, \ldots, N\}$, $h, \ell \in \{1, \ldots, k\}$, swapping the ℓ-th component gives the neighbor Y defined by

$$y_{rh} = \begin{cases} x_{rh} & \text{if } r \neq i, j \text{ or } h \neq \ell \\ x_{ih} & \text{if } r = j \text{ and } h = \ell \\ x_{jh} & \text{if } r = i \text{ and } h = \ell \end{cases} \tag{3.9}$$

We remark that, if Opt(D_1, J_1) be the optimality criterion, it perfectly makes sense to avoid considering pairs x_i and x_j such that $\mathcal{I}(X) \cap \{x_i, x_j\} = \emptyset$ since any swap involving two non-critical points cannot improve the D_1 value of the current LHD.
When Opt(ϕ) is adopted as optimality criterion, any exchange can, in general, lead to an improved value of ϕ. The RP local move for Opt(ϕ) optimality criterion is denoted by $\mathcal{LM}_{Rp\phi}$ and is equal to (3.9), the only difference being that we drop the requirement that at least one point must be critical.

We now propose some examples in order to illustrate the previously discussed local moves. We consider the $\mathcal{LM}_{RpD1}$ local move. Let a randomly generated initial solution be LHD A (see Figure 3.2(a)). Then a neighborhood solution of A, by considering points (1,2), (6,5) (here both are critical points), is LHD B, obtained after swapping the second coordinate of the points (1,2)

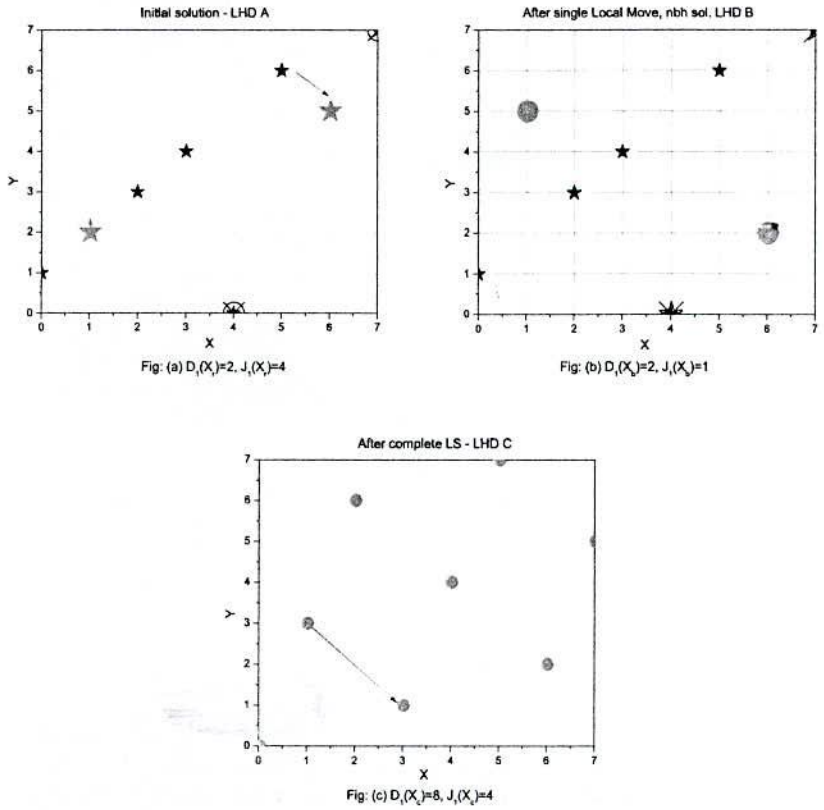

Fig. 3.2: Illustration of Neighborhood solutions for $\mathcal{LM}_{RpD1}$ based local search (LS) procedure

and (6,5) (See Figure 3.2 (b)). Also note that LHD B is an improving neighbor of LHD A, since $(D_1, J_1)(B) = (2,1)$ whereas $(D_1, J_1)(A) = (2,4)$. Note that by using two non-critical points like (4,0) and (7,7) of LHD A, there would be no chance to get an improving neighbor by means of a local move. Finally Figure 3.2 (c) shows the maximin LHD produced by the Local search procedure.

Acceptance Rule: We considered two types of acceptance rules, namely First Improve (FI) and Best Improve (BI) for each type of neighborhood. For the FI rule, the first improving neighbor whenever detected is adopted as starting point for a new neighborhood exploration. For the BI acceptance rule, the whole neighborhood of the current solution is searched for the *best* improving neighbor.

We warn again the reader that the meaning of "Y is better than X" can be defined accordingly with the $\text{Opt}(D_1, J_1)$ or $\text{Opt}(\phi)$ optimality criterion. So for the $\text{Opt}(D_1, J_1)$ optimality criterion: "Y is better than X" if

$$D_1(Y) > D_1(X) \quad \text{or} \quad (D_1(X) = D_1(Y) \ \text{ and } \ J_1(X) > J_1(Y)).$$

On the other hand for Opt(ϕ) optimality criterion : "Y is better than X" if

$$\phi_p(Y) < \phi_p(X),$$

where ϕ_p is defined by (3.5).

3.2.3 Perturbation Move ($\mathcal{P}_{\mathcal{M}}$)

Perturbation is the key operator in ILS, allowing the algorithm to explore the search space by jumping from one local optimum to another. Basically, a perturbation is *similar* to a local move, but it must be somehow *less local*, or, more precisely, it is a move within a neighborhood larger than the one employed in the local search. Actually the perturbation operator produces the initial solutions for all the local searches after the first one. Here we will propose mainly two types of perturbation operators namely 1. Cyclic Order Exchange (COE) and 2. Pairwise Crossover (PC).

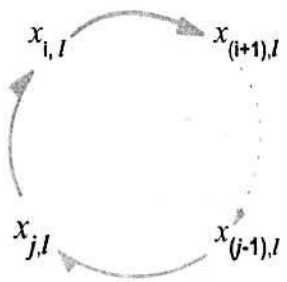

Fig. 3.3: Illustration of Cyclic Order Exchange perturbation technique

1. Cyclic Order Exchange (COE): Our first perturbation move procedure is Cyclic Order Exchange (COE). The operator COE produce a cyclic order exchange upon a randomly selected single component (column) of a randomly selected portion of the design points (rows). Here we present three variant of COE perturbation move techniques: Single Cyclic Order Exchange (SCOE) perturbation operation, Multiple Components Cyclic Order Exchange (MCCOE), and Multiple Single Cyclic Order Exchange (MSCOE).

1a. Single Cyclic Order Exchange (SCOE): For SCOE, we randomly choose two different rows (points), say x_i and x_j, such that $i < j$ and $j - i \geq 2$,

in the current LHD X^*. Then, we randomly choose a column (component), say ℓ. Finally, we swap in cyclic order the value of component ℓ from point x_i to point x_j — see Figure 3.3. The pseudo-code structure for SCOE is the following.

Step 1: `randomly select two different points` x_i `and` x_j
`such that` $i < j$ `and` $j - i \geq 2$
Step 2: `Randomly choose a component` ℓ
Step 3a: set temporarily $x^t_{j\ell} = x_{j\ell}$
for $t = j, j-1, \ldots, i-1$ **do**
  Step 3b: `Replace the component` $x_{(t)\ell}$ `by` $x_{(t-1)\ell}$
end for
Step 3c: and replace $x_{i\ell}$ *by* $x^t_{j\ell}$

Note that we require $j - i \geq 2$ because otherwise the perturbation would be a special case of the local move employed in the local search procedure. We illustrate the SCOE perturbation by an example. Assume we have the current LHD X^* with $N = 6$ and $k = 8$

$$X^* = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 5 & 4 \\ 1 & 2 & 3 & 4 & 5 & 0 & 4 & 3 \\ 2 & 3 & 4 & 5 & 0 & 1 & 3 & 2 \\ 3 & 4 & 5 & 0 & 1 & 2 & 2 & 1 \\ 4 & 5 & 0 & 1 & 2 & 3 & 1 & 0 \\ 5 & 0 & 1 & 2 & 3 & 4 & 0 & 5 \end{pmatrix} \tag{3.10}$$

Now we randomly choose two rows (points), say x_2 and x_5 and we randomly choose the column (component) $\ell = 4$. Then, after the SCOE perturbation we get the following LHD X' (bold faces denote the values modified with respect to X^*),

$$X' = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 5 & 4 \\ 1 & 2 & 3 & \mathbf{1} & 5 & 0 & 4 & 3 \\ 2 & 3 & 4 & \mathbf{4} & 0 & 1 & 3 & 2 \\ 3 & 4 & 5 & \mathbf{5} & 1 & 2 & 2 & 1 \\ 4 & 5 & 0 & \mathbf{0} & 2 & 3 & 1 & 0 \\ 5 & 0 & 1 & 2 & 3 & 4 & 0 & 5 \end{pmatrix} \tag{3.11}$$

Note that SCOE only slightly modifies the current LHD X^* but this exactly follows the spirit of ILS, where the perturbation should keep unchanged large portions of the current solution and should not completely disrupt its structure (see the discussion in Section 2.1).

1.b Multiple Components Cyclic Order Exchange (MCCOE): Our proposed MCCOE is a generalization of the single cyclic order exchange perturbation technique. In the MCCOE perturbation technique we perform SCOE operations on $s > 1$ columns instead of manipulating a single column.

For example, consider the previous LHD X^* defined in (3.10); we randomly fix a pair of rows (points), say x_2 and x_5 and we randomly choose the number of columns (components), say $s = 2$ in which the perturbation is performed. Then we randomly choose two columns say $\ell_1 = 4$ and $\ell_2 = 8$. Then, after

the MCCOE perturbation we get the following LHD X' (bold faces denote the modified values).

$$X' = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 5 & 4 \\ 1 & 2 & 3 & \mathbf{1} & 5 & 0 & 4 & \mathbf{0} \\ 2 & 3 & 4 & \mathbf{4} & 0 & 1 & 3 & \mathbf{3} \\ 3 & 4 & 5 & \mathbf{5} & 1 & 2 & 2 & \mathbf{2} \\ 4 & 5 & 0 & \mathbf{0} & 2 & 3 & 1 & \mathbf{1} \\ 5 & 0 & 1 & 2 & 3 & 4 & 0 & 5 \end{pmatrix} \tag{3.12}$$

1c. Multiple Single Cyclic Order Exchange (MSCOE): In this variant we perform the SCOE a random number of times rather than one time and each time we randomly select each SCOE operation. For illustration, consider the previous LHD X^* (3.10). In order to apply MSCOE we choose a number say $s = 2$ which indicate the number of times we perform SCOE operation. Then for the first SCOE we randomly choose a pair of rows (points), say x_2 and x_5 and a column, say $\ell_1 = 4$. Let X^t be the resulting LHD Now we perform the second SCOE: we again randomly choose a pair of rows(points), say x_1 and x_6, and a column, say $\ell_1 = 1$. Then after a SCOE perturbation on X^t we get the following final LHD X' (bold faces denote the values modified with respect to X^*),

$$X' = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} \mathbf{5} & 1 & 2 & 3 & 4 & 5 & 5 & 4 \\ \mathbf{0} & 2 & 3 & \mathbf{1} & 5 & 0 & 4 & 3 \\ \mathbf{1} & 3 & 4 & \mathbf{4} & 0 & 1 & 3 & 2 \\ \mathbf{2} & 4 & 5 & \mathbf{5} & 1 & 2 & 2 & 1 \\ \mathbf{3} & 5 & 0 & \mathbf{0} & 2 & 3 & 1 & 0 \\ \mathbf{4} & 0 & 1 & 2 & 3 & 4 & 0 & 5 \end{pmatrix} \tag{3.13}$$

2. Pairwise Crossover: The second type of perturbation move that we consider is the Pairwise Crossover (PC). It is similar to biological crossover – we randomly select two points (rows) and then randomly selected portions of them which are interchanged. Here we propose three variant of PC namely Single Pair Crossover (SPC), Multiple Pair Crossover (MPC) and Critical-point Far-most Pair Crossover (CFPC).

2a. Single Pair Crossover (SPC): For SPC, we first randomly select two rows, say, x_i and x_j , $i \neq j$, in the current LHD X^*; then we randomly select a component, say $\ell \geq 2$. Finally all the components $1, \ldots, \ell$ of x_i are swapped with the corresponding components of x_j — refer to Figure 3.4. Note that we require $\ell \geq 2$, since otherwise it would be a single local move. It is also worthwhile to remark that the PC perturbation is meaningful only when number of factors of the LHD is grater than three. The pseudo code structure of SPC is as follows:

Step 1: `randomly select two different points` x_i `and` x_j `such that` $i \neq j$
Step 2: `Randomly choose a component` ℓ `such that` $\ell \geq 2$
for $k = 1, \ldots, \ell$ **do**
  Step 3: swap(x_{ik}, x_{jk})

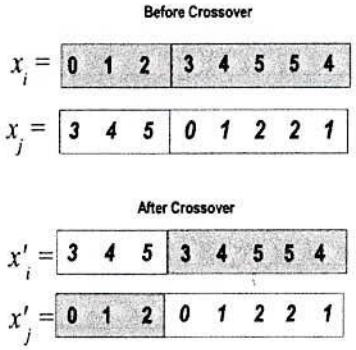

Fig. 3.4: Illustration of Single Pair Crossover perturbation technique

end for

2b. Multiple Pairwise Crossover (MPC): MPC is a generalization of the SPC perturbation technique. In the MPC perturbation technique we randomly choose multiple pairs of points rather than a single pair of points to perform crossover.

It is worthwhile to remark that the randomly generated integer value s (how many times SPC is performed in MPC) should be a small value, since if s be a large number, the produced MPC-perturbed LHD could not preserve much of the structure of the current local optimum. That is, MPC would become (practically) a random initialization procedure. For illustration of MPC perturbation moves, consider the LHD X^* given by Eq. (3.10). Now we choose a number say $s = 2$ which indicate the number of times we perform SPC operations. Then for the first SPC we randomly choose a pair of rows (points), say x_2 and x_5, and randomly fix a column, say $\ell_1 = 4$. After the SPC perturbation, we get the following LHD X^t (bold faces denote modified values).

$$X^t = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 5 & 4 \\ \mathbf{4} & \mathbf{5} & \mathbf{0} & \mathbf{1} & 5 & 0 & 4 & 3 \\ 2 & 3 & 4 & 5 & 0 & 1 & 3 & 2 \\ 3 & 4 & 5 & 0 & 1 & 2 & 2 & 1 \\ \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & 2 & 3 & 1 & 0 \\ 5 & 0 & 1 & 2 & 3 & 4 & 0 & 5 \end{pmatrix} \tag{3.14}$$

Now we perform the second SPC: we again randomly choose a pair of rows (points), say x_2 and x_6, and randomly fix a column, say $\ell_2 = 2$. Then after another SPC perturbation on X^t we get the following final LHD X' (bold faces denote the values modified with respect to X^*),

$$X' = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 5 & 4 \\ \mathbf{5} & \mathbf{0} & \mathbf{0} & \mathbf{1} & 5 & 0 & 4 & 3 \\ 2 & 3 & 4 & 5 & 0 & 1 & 3 & 2 \\ 3 & 4 & 5 & 0 & 1 & 2 & 2 & 1 \\ \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & 2 & 3 & 1 & 0 \\ 4 & \mathbf{5} & 1 & 2 & 3 & 4 & 0 & 5 \end{pmatrix} \tag{3.15}$$

It is important to remark that when $s > 1$, MPC may also produce a so-called `dead offspring` — by this we mean the case where $X' = X^*$. An algorithm incorporating MPC should detect such events.

2c. Critical-point Far-most Pair Crossover (CFPC): A point x_i is randomly selected, and the mate point x_j involved in the crossover is selected so that x_j has the maximum possible distance from x_i (with ties broken randomly). Both SPC and MPC can then be applied.

3.2.4 Stopping Rule ($\mathcal{S}_\mathcal{R}$)

We use a very simple stopping Rule ($\mathcal{S}_\mathcal{R}$). We introduce an integer parameter called `MaxNonImp`, and the algorithm will stop if the currently best local optimizer X^* cannot be improved for `MaxNonImp` consecutive perturbations.

4. COMPUTATIONAL EXPERIMENTS AND DISCUSSION ABOUT ILS WITH OPT(D_1, J_1)

In Section 3.2 we have proposed our algorithms and also have discussed details about them. Here we will report on experiments performed in order to investigate the different components and their impact on the efficiency of the algorithms. Since we have proposed algorithms *driven* by two different optimality criteria, at first we will perform experiments for the Opt(D_1, J_1) optimality criterion, and in the next chapter for the Opt(ϕ) optimality criterion. We also perform some experiments to compare the results with those available in the literature. Since the first operator of our proposed ILS algorithms is `Initialization` and it is just simple random permutation on $\{0, 1, 2 \ldots, N-1\}$, we will perform experiments to investigate the performance of other operators like Local search procedure, Perturbation moves procedure and so on.

4.1 Experiments on Local Search procedure

Local search is the first relevant operator of our proposed ILS based algorithms. In order to appreciate the importance of this operator we will first make a comparison between the simplest Random Search (RS) method, in which there is no local search operator, and the simplest method based on multiple local searches, Multistart (MS), in which only local searches are performed. Moreover, we will study the impact of different possible choices in the definition of the local search procedure.

4.1.1 Impact of LocalSearch

Our first goal is to assess that using a local search — and hence focusing only on local optima during the overall search — is a profitable choice. Thus we compare a trivial Random Search (RS) with a Multistart (MS) procedure equipped with a local search. We consider LHDs with $N = 3, 4, \ldots, 25$ and $k = 3, 5, 7, 10$; we do not report results for higher values of N, as RS is — quite expectedly — strongly dominated by MS as N grows. For the MS based approach we adopted a RP local move with FI acceptance rule based Local search procedure.

We set the number of runs for the MS approach to $R = 1000$ for all (N, k) considered pairs, whereas we ran the RS approach with 50 times more runs than the MS approach for each (N, k) considered. In order to investigate the impact of LocalSearch operator, we compare the performance of the RS and MS methods. We plotted the N values on the x-axis and the percentage improvements (in terms of Maximin (Mm) values) of MS over RS on y-axis, for each $k = 3, 5, 7, 10$ (see Figure 4.1). We observe in the figure that the performance of the MS approach (in percentage) is approximately linearly increasing with N for all k values, and RS is dominated.

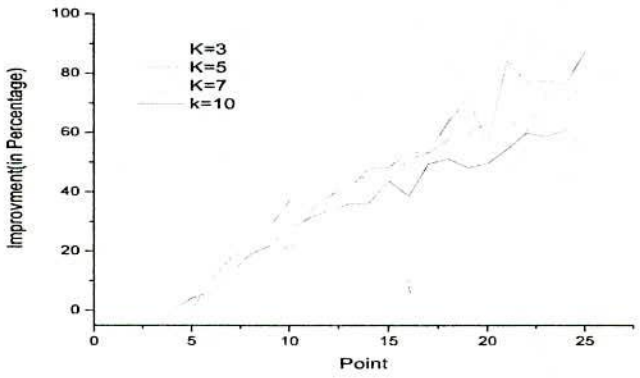

Fig. 4.1: Comparison of the performance of MS with that of RS (in percentage)

This phenomenon is somehow expected, because the search space for RS comprises every feasible solution, so it is very large and its cardinality drastically grows with N like $(N!)^{k-1}$. So RS has very few chances of finding "good" LHDs. On the other hand the search space of MS is a relatively small subset of the whole set of feasible solutions, composed of local optima.

4.1.2 Impact of RP and CP Local moves

Since we have proposed two different types of local search moves — namely Row-wise pairwise (RP) local moves and Column-wise pairwise (CP) local moves — we performed experiments to investigate the impact of each type of local move.

For the experiments we consider LHDs with $N = \{1, 2, \ldots, 25, 5i\}$ where $i = 6, 7, \ldots, 20$, and $k = 3, 5, 7, 10$. We tested on each LHD a multistart (MS) procedure for a number of runs $R = 1000$; in local search, we adopted the first improvement (FI) acceptance rule for both RP and CP local moves. In Table 4.1 we report the best LHD values (Mm values) over 1000 runs. We observe in the table that for $k = 3$, the RP based Local search generated a better LHD in 13 cases, whereas the CP based Local search is the winner in 5 cases. On the other hand for $k = 10$, RP works better in 14 cases and CP in 16 cases. We also observe that the total computational cost of the two approaches is quite similar, see Table 4.1.

We may conclude that there is no significant difference between the performances of RP and CP based local search procedure.

4.1.3 Impact of FI and BI Acceptance rule

In Section 3.2 we have proposed two type of acceptance rules, namely FI and BI, for the local search procedure; in this section we report about some experiments whose goal is investigating the impact of these acceptance strategies. Since we investigate the impact of acceptance strategy on Local Search techniques.

Tab. 4.1: The Comparison of the performance of RP and CP with FI acceptance rule based Local search procedures

N	$k = 3$		$k = 5$		$k = 7$		$k = 10$	
	RP	CP	RP	CP	RP	CP	RP	CP
3	6	6	8	8	13	13	19	19
4	6	6	14	14	21	21	33	33
5	11	11	24	24	32	32	50	50
6	14	14	32	32	47	47	68	68
7	17	17	40	39	61	61	90	89
8	21	21	50	50	77	78	115	113
9	22	22	59	61	91	92	140	140
10	27	27	78	78	108	107	169	169
11	30	30	78	78	125	127	201	200
12	36	36	88	89	145	145	227	226
13	41	41	99	100	169	167	259	258
14	41	41	110	111	189	188	295	296
15	48	43	123	122	209	207	336	334
15	48	46	138	135	231	230	373	370
17	53	51	147	148	252	255	411	412
18	54	54	159	161	280	277	453	454
19	56	57	171	174	304	301	501	499
20	61	61	186	187	330	327	541	541
21	65	65	199	200	356	355	590	590
22	69	69	214	215	384	381	638	650
23	72	72	232	229	413	411	686	690
24	76	75	244	243	442	443	738	741
25	78	77	257	256	477	472	791	798
30	94	94	319	322	615	620	1063	1072
35	116	113	402	410	785	786	1379	1386
40	134	133	499	495	983	967	1739	1731
45	153	158	597	590	1183	1196	2126	2115
50	174	179	699	693	1392	1389	2550	2548
55	202	202	805	795	1608	1640	2987	3003
60	226	221	914	911	1866	1853	3483	3495
65	248	245	1035	1029	2134	2106	3975	3988
70	276	270	1153	1157	2409	2430	4524	4530
75	293	290	1278	1275	2692	2695	5096	5119
80	317	315	1398	1409	2989	2978	5742	5687
85	344	344	1542	1549	3285	3292	6315	6357
90	372	369	1673	1681	3606	3618	7014	6992
95	398	401	1827	1836	3967	3926	7687	7741
100	420	422	1981	1981	4313	4267	8406	8394
No. of Bet. LHD	13	5	12	17	19	13	14	16
T. Time (in sec)	576	558	1661	1676	3501	3506	8919	8600

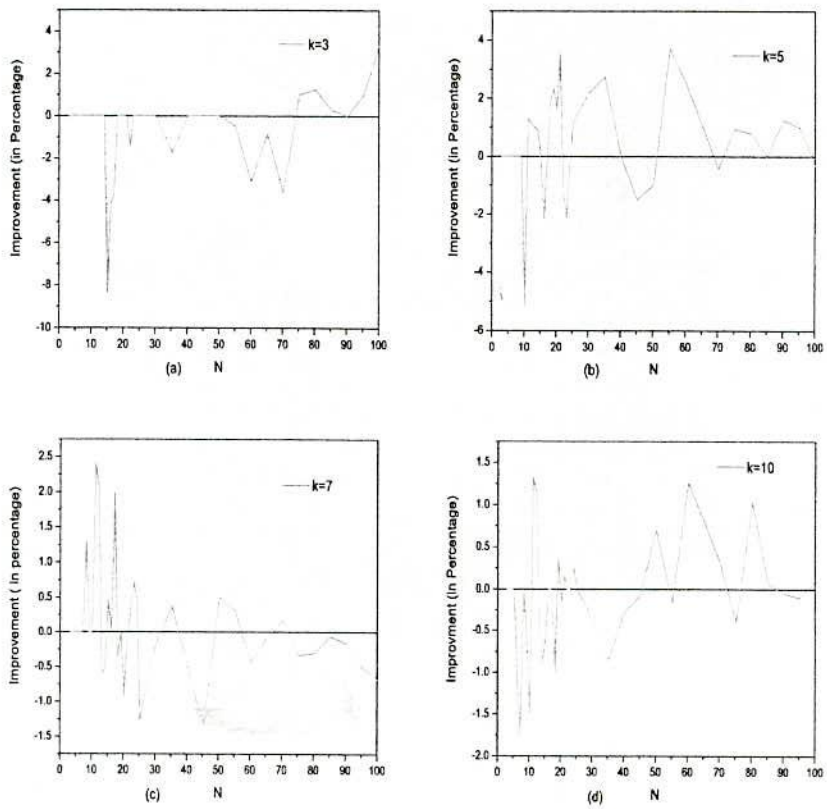

Fig. 4.2: Comparison of the performance of BI with that of FI (in percentage)

RP local moves; all other settings for the experiments are the same described in the previous sections. We ran the MS procedure with FI and BI acceptance rules respectively for all the LHDs defined above.

For the investigation of the impact of acceptance criterion, we plotted on x-axis the N values, and on y-axis the percentage improvements in terms of Mm values of the BI based approach w.r.t. the FI based approach for each $k = 3, 5, 7, 10$ — see Figure 4.2. Though it is not very clear from the figure, for lower values of N the FI acceptance strategy performs slightly better for all k except $k = 7$. On the other hand for large N values, the BI acceptance strategy performs slightly better for all k except $k = 7$. By looking at the overall results we conclude that, from the point of view of solution quality, there is no relevant evidence of one acceptance rule dominating the other.

Anyway we observe in Figure 4.3 that the FI acceptance strategy based algorithm is always computationally cheaper than that of BI acceptance strategy based algorithm. The BI based algorithm needs more CPU time because a complete neighborhood is explored for each visited solution, whereas with the FI acceptance rule only a partial exploration in most of the neighborhoods is

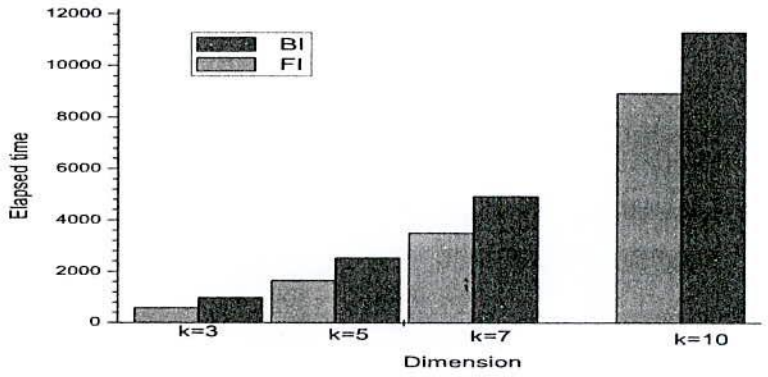

Fig. 4.3: Comparison of the elapsed time of FI and BI acceptance strategy based algorithm (in second)

performed.

4.2 Experiments on perturbation moves

4.2.1 Impact of Perturbation Operator

In order to assess the contribution of the perturbation operator to the performances of the ILS algorithm, we compare ILS with a MS algorithm. We stress that using the perturbation operator for starting a new local search from a previously reached local optimum is the key feature distinguishing ILS from MS — in MS the new starting point is drawn at random.

For this set of experiments we consider the LHDs with $N = 3, 4, \ldots, 25$ and $k = 3, 5, 7, 10$ (we do not report results for higher values of N, but remark that MS becomes strongly dominated by ILS on such tests). In both the MS and ILS procedures we use a Local Search equipped by the RP local move with FI acceptance rule. In the ILS based procedure we use SCOE perturbation moves procedure with `MaxNonImp` =1000 and number of runs for each (N, k) set to $R = 100$. In order to have a fair comparison, on each (N, k) test we first ran the ILS based method and count the total number of local searches, call it $L(N, k)$, needed for optimized LHD with each pair value of (N, k). Then we ran MS with $L(N, k)$ number of trials for the corresponding (N, k) pairs.

We plotted on the x-axis the N values, and on y-axis the percentage improvements of Mm values obtained by ILS compared with those given by MS, for each $k = \{3, 5, 7, 10\}$ (see Figure 4.4). We notice from the figure that ILS always has a performance at least as good as the one of MS and, in particular for large k values, the improvement guaranteed by ILS tends to increase with the number N of points. The cause of this fact is that each new starting point of MS is drawn at random and so it usually fails to preserve any "good" part of the previous locally optimal LHD. On the other hand, the starting points of the ILS based method are generated by applying to each local optimum a per-

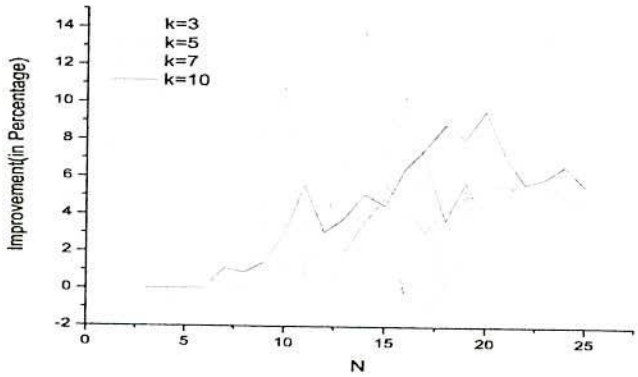

Fig. 4.4: Comparison of the performance of ILS with that of MS (in percentage)

Tab. 4.2: The Comparison of the computational cost of MS and ILS (D_1, J_1)(time is in second)

k	MS	ILS
3	811	**416**
5	2179	**1114**
7	4211	**2117**
10	9589	**4796**
Total time	16790	**8443**

turbation operation, which is slightly larger than a local move. The perturbed local optimum usually preserves a good part of the configuration of the previous locally optimal LHD and at the same time allows an effective exploration of the search space.

Table 4.2 shows the elapsed time of MS and ILS based methods for each k value of the above described experiments. We observe that, although we allowed the same number of local searches for both ILS and MS, the running time for ILS was clearly inferior. Again, this is due to the fact that the starting points of local searches in ILS are only slight perturbations of local minimizers — with an already partially-optimized structure. Hence the local search procedure (often) reaches a new local optimum in fewer iterations with respect to the case of MS where starting points are completely random ones.

In order to offer more details on how MS and ILS differ in their behaviors, we performed further experiments. With the same parameter setting, we first ran ILS with only one trial for $(N, k) = (17, 3)$ and then ran MS with same number of trial as the number of local searches needed by ILS. We display the partial search history of initial solutions of MS (where initial solutions are generated randomly) and ILS (where the initial solutions are generated by the perturbation operator from local optimizers except the first one) approaches in Figure 4.5. From this figure we notice that the initial solutions of ILS are

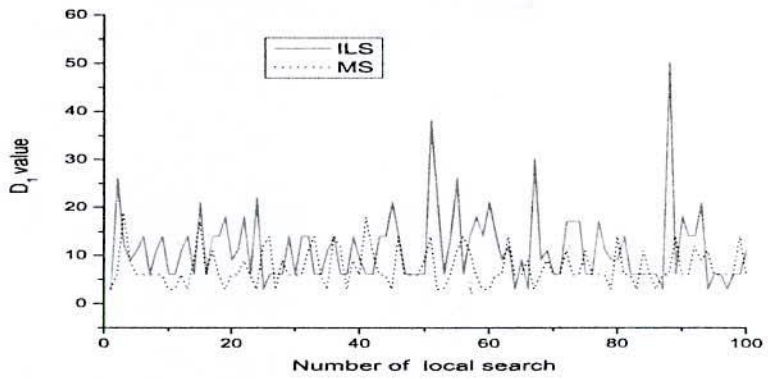

Fig. 4.5: Comparison of search history of initial solutions between MS and ILS for $(N, k) = (17, 3)$ (partial search space)

almost always better than those of MS.

Also, we display the partial search history of locally optimal solutions delivered by MS and ILS in Figure 4.6. We observe from Figure 4.6 that in almost all the local searches ILS produces better local optimizers than MS.

Figure 4.7 shows the partial search history of MS including both initial solutions and corresponding locally optimal solutions for $(N, k) = (17, 3)$, and Figure 4.8 displays the same data collected for ILS. In both figures red balls denote initial solutions and black squares indicate locally optimal solutions.

We can observe in Figure 4.7 that many initial solutions have small objective values (less than 15) and the corresponding local optimal solution also have relatively small values (less than 45). On the other hand we observe in Figure 4.8 that a large amount of initial solutions have relatively large values (greater than 15) and corresponding local optimal solutions also have relatively large values (greater than 45). That is better initial solutions almost always produce relatively better solutions. But in Figure 4.8 we also observe that though, at local search step 735 of ILS approach, the initial solution is a good one, it produces a not so good local optimal solution. In spite of this counterexample, it is worthwhile to remark that though there is no guarantee that better initial solutions always provide better local solutions, this is, in fact, often the case.

Another observation at step 725 of the local search of both MS and ILS approach: the initial solution of ILS is relatively worse than that of MS (see in Figures 4.7 and 4.8 the large blue circles, the initial solution of MS approach has value 11 whereas the initial solution of ILS approach has value 6) but the local optimal solution delivered by ILS is relatively better (it has value 42 whereas

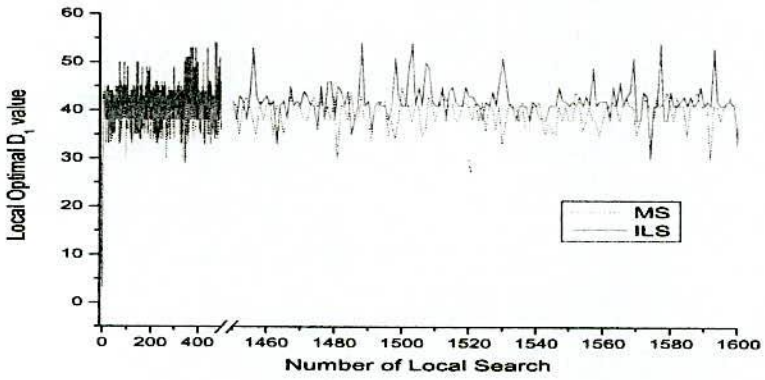

Fig. 4.6: Comparison of search history of local optimal solutions between MS and ILS for $(N, k) = (17, 3)$ (partial search space)

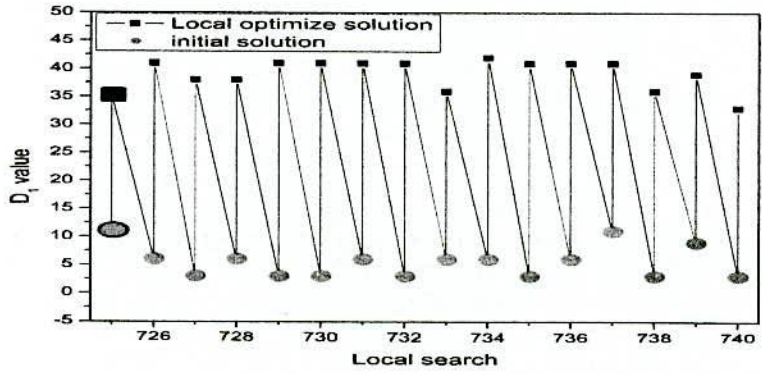

Fig. 4.7: The partial Search history of MS local search including initial solutions and optimal solutions for $(N, k) = (17, 3)$

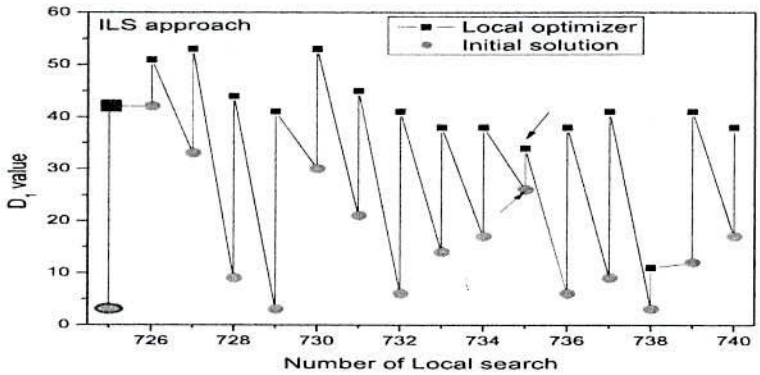

Fig. 4.8: The partial Search history of ILS local search including initial solutions and local optimizer for $(N, k) = (17, 3)$

the local optimal solution detected by MS is 35). We might conjecture that in this case the perturbation move has considerably worsened a small portion of the current solution, but, being such portion quite limited, the local search procedure is able to "repair" it quite effectively thus leading to a good local optimum.

4.2.2 *Impact of different perturbation moves*

We have proposed mainly two type of Perturbation moves procedures — COE and PC perturbation moves — and for each of them several variants; in this section we will investigate their performances. For all the experiments considered in this section, we used LHDs with $N = \{1, 2, \ldots, 25, 5i\}$ where $i = 6, 7, \ldots, 20$, and $k = 3, 5, 7, 10$. We consider RP local moves with BI acceptance rule based local search for all the approaches considered. We also set MaxNonImp = 1000, and number of runs $R = 10$ for all the tests.

We first present experiments about the impact of the three variants of COE perturbation moves procedures. In the table 4.3 we report the best Mm values (over 10 runs). In the table "B/E LHD " means total number of better or equal LHD values (with respect to this comparison) obtained for each k dimension for each version. We observe that the ILS algorithm equipped with the SCOE perturbation outperforms the other two ILSs. ILS with MSCOE perturbations seems to perform somehow better than ILS with MCCOE, but worse than ILS with SCOE. Although ILS with the MCCOE perturbation is relatively cheaper, it frequently obtains worse Mm values with respect to the other two tested perturbations. Hence we recommend the SCOE perturbation as the best choice among the three versions of COE perturbation moves.

Our next experiment is about the impact of the three variants of PC per-

Tab. 4.3: Comparison among different COE perturbation move procedures. Note that in the table SCOE, MCCOE and MSCOE are denoted as SC, MCC and MSC.

	$k = 3$			$k = 5$			$k = 7$			$k = 10$		
N	SC	MCC	MSC	SC	MCC	MSC	SC	MCC	MSC	SC	MCC	MSC
3	6	6	6	8	8	8	13	13	13	19	19	19
4	6	6	6	14	14	14	21	21	21	33	33	33
5	11	11	11	24	24	24	32	32	32	50	50	50
6	14	14	14	32	32	32	46	47	47	67	68	68
7	17	17	17	38	38	40	60	60	61	89	89	89
8	21	21	21	50	50	50	78	77	79	114	113	114
9	22	22	22	61	59	60	93	92	93	141	140	140
10	26	27	27	78	76	78	108	108	108	172	171	172
11	29	30	30	79	79	79	128	128	129	206	206	206
12	33	33	33	88	89	91	152	150	151	234	232	234
13	41	41	41	102	101	102	179	173	176	271	266	269
14	42	41	41	114	111	113	215	207	210	306	302	304
15	48	43	45	128	123	125	223	219	215	343	340	342
15	49	46	49	141	134	144	241	238	239	390	384	384
17	51	51	51	154	148	154	262	256	259	431	417	432
18	54	54	56	166	159	165	285	280	284	482	471	480
19	59	56	57	181	176	178	315	307	313	539	532	527
20	62	61	62	198	194	198	345	331	343	572	560	602
21	66	65	66	215	200	213	372	364	370	630	600	625
22	69	66	70	227	214	222	406	387	394	669	660	668
23	73	70	73	246	228	237	438	415	427	722	698	713
24	77	75	76	255	242	250	465	449	460	785	760	766
25	81	77	81	272	259	267	497	473	491	832	809	822
30	101	97	101	356	337	347	667	644	656	1136	1108	1130
35	122	121	122	454	426	449	856	815	854	1491	1435	1467
40	149	138	146	552	525	549	1055	1005	1055	1894	1804	1850
45	166	158	168	662	611	658	1284	1226	1258	2315	2238	2270
50	189	181	190	755	719	764	1523	1440	1500	2770	2656	2749
55	216	206	211	890	832	867	1781	1682	1754	3293	3133	3235
60	241	230	234	998	952	979	2028	1943	1999	3778	3646	3749
65	266	250	261	1123	1067	1115	2340	2213	2318	4383	4195	4358
70	290	281	289	1262	1182	1235	2637	2475	2580	4915	4750	4853
75	321	302	321	1404	1307	1387	2944	2769	2866	5644	5347	5504
80	344	333	345	1548	1434	1514	3286	3084	3168	6252	5985	6143
85	374	354	371	1679	1591	1667	3588	3399	3551	6945	6646	6813
90	394	378	404	1837	1715	1796	3934	3720	3899	7659	7340	7505
95	420	406	433	1994	1867	1936	4355	4082	4208	8450	8037	8333
100	454	427	453	2173	2004	2135	4676	4448	4667	9201	8749	9098
B/E LHD	29	12	27	33	5	14	34	5	10	35	6	10
Tot. Time (hrs)	1.25	0.84	1.12	3.49	2.14	3.27	7.55	4.56	6.4	17.92	11.39	16.33

Tab. 4.4: Comparison among different PC perturbation move procedures

	$k=3$			$k=5$			$k=7$			$k=10$		
N	SPC	MPC	CFPC	SPC	MPC	CFPC	SPC	MPC	CFPC	SPC	MPC	CFPC
3	6	6	6	8	8	8	13	13	13	19	19	19
4	6	6	6	14	14	14	21	21	21	33	33	33
5	11	11	11	24	24	24	32	32	32	50	50	50
6	14	14	14	32	32	32	46	47	45	67	67	67
7	17	17	17	38	39	38	60	61	60	90	89	89
8	21	21	19	48	50	48	77	78	77	113	114	114
9	22	22	22	59	61	59	90	93	91	140	141	141
10	27	26	26	76	78	76	108	111	108	170	173	172
11	30	30	29	80	80	78	125	129	129	205	208	206
12	36	33	33	88	91	88	149	152	147	234	235	234
13	41	41	36	100	102	100	176	178	175	266	271	270
14	42	41	41	111	116	113	206	210	197	304	303	299
15	44	44	44	127	128	122	217	219	214	345	342	343
15	49	50	46	139	146	136	237	238	239	391	387	387
17	51	53	51	151	156	152	261	263	259	435	433	427
18	54	54	54	165	168	166	287	291	284	488	477	476
19	56	59	57	181	179	181	311	314	317	525	518	518
20	62	62	59	199	199	195	340	347	340	575	586	571
21	66	66	66	211	212	210	369	374	370	631	617	615
22	70	69	69	230	232	225	401	406	401	674	668	664
23	74	74	73	242	239	238	437	435	434	721	724	716
24	77	77	77	254	258	251	466	463	467	777	779	780
25	84	84	81	274	269	271	496	498	500	831	843	827
30	105	102	101	353	355	363	675	676	663	1150	1145	1133
35	125	129	123	459	458	454	862	866	865	1494	1504	1496
40	146	149	145	556	563	553	1080	1071	1059	1907	1894	1889
45	174	173	171	663	674	648	1311	1299	1290	2312	2330	2297
50	200	197	189	781	784	768	1540	1548	1535	2806	2783	2781
55	224	225	216	908	911	890	1802	1799	1782	3310	3269	3295
60	254	246	245	1017	1011	1021	2085	2093	2057	3868	3797	3813
65	278	278	266	1163	1173	1136	2363	2359	2345	4407	4398	4391
70	307	301	299	1294	1291	1278	2683	2681	2638	5022	5027	5017
75	338	330	325	1427	1435	1419	2972	2992	2981	5639	5610	5663
80	369	354	347	1596	1578	1576	3326	3343	3302	6369	6331	6320
85	393	389	377	1723	1752	1716	3639	3693	3628	7027	6997	6971
90	420	420	411	1908	1894	1882	4048	4052	3994	7782	7738	7680
95	460	453	434	2067	2056	2031	4408	4360	4368	8592	8528	8473
100	483	481	474	2223	2235	2171	4770	4787	4814	9387	8749	9235
B/E LHD	32	24	10	15	28	6	10	28	7	24	16	8
Tot. Time (hrs)	0.90	0.73	0.47	2.78	3.05	1.65	5.26	6.80	3.90	13.03	14.81	10.32

turbation moves procedures. The results of the tests are reported in the table 4.4. The meaning of "B/E LHD " in the table is the same discussed above. We observe that the performances of ILS with SPC and MPC perturbations are comparable whereas ILS with FCPC seems worse than the other two approaches. We also observe that the computational cost of SPC and MPC based approach are comparable; FCPC is cheaper but most of the times it produces worse LHDs with respect to the other two versions of PC perturbations. Hence we recommend SPC as the best choice among the three version of the PC perturbation moves considered.

We finally wish to offer a comparison between COE and PC perturbations. For this comparison we consider SCOE and SPC, that turn out to be the best option for the two techniques. The comparison of the two approaches is displayed in the table 4.5. We comment on the results by splitting the table in two parts: the label "a " denotes the first section of the table, i.e. the tests for $3 \leq N \leq 25$ and "b " denotes the last part of the table, with tests for $N > 25$. In the row named "Bet LHD (a,b) " : "a " and "b " denotes the total number of better (by the comparison of the two versions considered) LHD values in section "a " and "b " respectively for each k value. We notice from the table that for small values of N, the SCOE perturbation works slightly better than SPC perturbation based approach except for $k = 10$, whereas for large values of N the results are reversed, and the SPC perturbation performs better than SCOE.

Tab. 4.5: Comparison between COE and SPC perturbation move procedures

N	$k = 3$		$k = 5$		$k = 7$		$k = 10$	
a	SCOE	SPC	SCOE	SPC	SCOE	SPC	SCOE	SPC
3	6	6	8	8	13	13	19	19
4	6	6	14	14	21	21	33	33
5	11	11	24	24	32	32	50	50
6	14	14	32	32	46	46	67	67
7	17	17	38	38	60	60	89	90
8	21	21	50	48	78	77	114	113
9	22	22	61	59	93	90	141	140
10	26	27	78	76	108	108	172	170
11	29	30	79	80	128	125	206	205
12	33	36	88	88	152	149	234	234
13	41	41	102	100	179	176	271	266
14	42	42	114	111	215	206	306	304
15	48	44	128	127	223	217	343	345
15	49	49	141	139	241	237	390	391
17	51	51	154	151	262	261	431	435
18	54	54	166	165	285	287	482	488
19	59	56	181	181	315	311	539	525
20	62	62	198	199	345	340	572	575
21	66	66	215	211	372	369	630	631
22	69	70	227	230	406	401	669	674
23	73	74	246	242	438	437	722	721
24	77	77	255	254	465	466	785	777
25	81	84	272	274	497	496	832	831
b								
30	101	105	356	353	667	675	1136	1150
35	122	125	454	459	856	862	1491	1494
40	149	146	552	556	1055	1080	1894	1907
45	166	174	662	663	1284	1311	2315	2312
50	189	200	755	781	1523	1540	2770	2806
55	216	224	890	908	1781	1802	3293	3310
60	241	254	998	1017	2028	2085	3778	3868
65	266	278	1123	1163	2340	2363	4383	4407
70	290	307	1262	1294	2637	2683	4915	5022
75	321	338	1404	1427	2944	2972	5644	5639
80	344	369	1548	1596	3286	3326	6252	6369
85	374	393	1679	1723	3588	3639	6945	7027
90	394	420	1837	1908	3934	4048	7659	7782
95	420	460	1994	2067	4355	4408	8450	8592
100	454	483	2173	2223	4676	4770	9201	9387
Bst LHD (a,b)=(23, 15)	(2,1)	(6,14)	(12,1)	(4,14)	(15,0)	(2,15)	(10,2)	(10,13)
Tot. Time (hrs)	1.25	0.90	3.49	2.78	7.55	5.26	17.92	13.03

Also SPC seems to be relatively cheaper than SCOE. How can this behavior be explained? If we recall the details of the two perturbation techniques (see section 3.2, we can notice that SCOE perturbs more than one point but only by a single factor. On the other hand SPC perturbs only a pair of points but by more than one factor of that pair.

A possible explanation for these results is the following. We already stressed that the *size* of the perturbation is crucial in order to get a good exploration of the search space: an excessively small size may prevent the exploration of different local optima, while an excessively large perturbation would simply result in a multistart search. For large instances, the perturbation offered by SCOE may be excessively small, whereas SPC gets larger changes in the current local optimum. On the other hand, on small instances, SPC may give excessively large perturbations, downgrading the performances toward those of a multistart search. It can be that, on larger instances, even SPC is not sufficient and larger perturbations — for example, MPC — should be used.

4.3 *Impact of Stopping Rule*

We have already discussed in Section 3.2 about the stopping condition used in the proposed algorithms. This is driven by a simple integer parameter called `MaxNonImp` (MNI): how long the perturbation operations (and subsequent local searches) are performed is defined by this parameter.

The role of the MaxNonImp parameter is crucial for the effectiveness of the al-

Tab. 4.6: The average performance of ILS based method for different MaxNonImp value when $k = 3$

N	MNI-100	MNI-500	MNI-1000	MNI-2000	MNI-3000	MNI-4000	MNI-6000	MNI-8000	MNI-10000
6	**14**	14	14	14	14	14	14	14	14
8	**20**	20	20	20	20	20	20	20	20
10	**25**	25	25	25	25	25	25	25	25
12	32	**33**	33	33	33	33	33	33	33
14	**40**	40	40	40	40	40	40	40	40
16	44	**46**	46	46	46	46	46	46	46
18	52	**53**	53	53	53	53	53	53	53
20	57	59	**61**	61	61	61	61	61	61
22	65	**68**	68	68	68	68	68	68	68
24	71	74	**75**	75	75	75	75	75	75
26	80	82	83	**84**	84	84	84	84	84
28	87	**90**	90	90	90	90	90	90	90
30	96	98	**99**	99	99	99	99	99	99
32	103	106	106	**107**	107	107	107	107	107
34	111	115	**117**	117	117	117	117	117	117
36	118	122	124	**125**	125	126	126	126	126
38	126	131	132	134	135	135	**136**	136	136
40	135	139	142	142	143	143	143	**144**	144
42	141	147	149	150	150	**152**	152	152	152
44	153	157	159	160	160	161	**162**	162	162
46	159	164	168	170	170	170	172	**173**	173
48	167	176	178	179	180	180	**181**	181	181
50	178	185	188	190	191	**192**	192	192	192

gorithms: an excessively small number of perturbations usually results in worse solutions delivered, but an excessively large number of perturbations results in a waste of CPU time. We believe that finding a "perfect" `MaxNoImp` value is almost impossible, because of the random nature of some components of the algorithm. This is why we would like to find out approximate `MaxNonImp` values for the proposed ILS algorithm. Since in LHD there are two key parameters — number of design points (N) and number of factors (k) — we want to investigate the impact of N and k on the most appropriate value of the `MaxNonImp` parameter. For this purpose we performed the experiments reported below. For all the experiments described in this section, we have the following parameter setting: Acceptance rule= Best Improve(BI), LocalSearch=RP; Perturbation Technique=SCOE and MaxNonImp=1000.

4.3.1 *Impact of N on MaxNonImp*

In order to investigate the impact of N, we consider the LHDs with $k = 3, 5, 7, 10$ and $N = 2i : i = 1, 2, \ldots, 25$. We set `MaxNonImp`=10,000 as a stopping condition and number of runs $R = 10$. For each run with `MaxNonImp` =10,000, we also store (during the same run) the best results that would be obtained if `MaxNonImp`=100, 1000, 2500, 2500, 5000, 8000, and 10,000 as well.

At first we report the maximin LHDs in Tables 4.6-4.9 for all the tested

Tab. 4.7: The average performance of ILS based method for different MaxNonImp value when $k = 5$

N	MNI-100	MNI-500	MNI-1000	MNI-2000	MNI-3000	MNI-4000	MNI-6000	MNI-8000	MNI-10000
6	**32**	32	32	32	32	32	32	32	32
8	**48**	48	48	48	48	48	48	48	48
10	73	**76**	76	76	76	76	76	76	76
12	87	**88**	88	88	88	88	88	88	88
14	108	110	110	**111**	111	111	111	111	111
16	133	**135**	135	135	135	135	135	135	135
18	158	163	164	164	**165**	165	165	165	165
20	185	189	**191**	191	191	191	191	191	191
22	213	217	219	**221**	221	221	221	221	221
24	241	248	249	250	**251**	251	251	251	251
26	270	278	280	280	282	**283**	283	283	283
28	297	307	310	312	**314**	314	314	314	314
30	331	343	349	350	350	**351**	351	351	351
32	366	377	379	382	383	383	383	**385**	385
34	402	412	418	423	424	424	425	**426**	426
36	441	452	457	461	464	**465**	465	465	465
38	469	490	494	500	501	502	**503**	503	503
40	514	531	534	538	541	542	546	546	**547**
42	553	564	568	578	582	586	587	**589**	589
44	593	607	613	618	619	621	622	**623**	623
46	631	656	663	670	670	672	673	675	**676**
48	672	699	707	714	714	716	718	720	**722**
50	713	741	749	753	755	758	761	763	**765**

Tab. 4.8: The average performance of ILS based method for different MaxNonImp value when $k = 7$

N	MNI-100	MNI-500	MNI-1000	MNI-2000	MNI-3000	MNI-4000	MNI-6000	MNI-8000	MNI-10000
6	45	**46**	46	46	46	46	46	46	46
8	76	**77**	77	77	77	77	77	77	77
10	106	107	**108**	108	108	108	108	108	108
12	146	147	**148**	148	148	148	148	148	148
14	188	**199**	199	199	199	199	199	199	199
16	230	234	235	235	**236**	236	236	236	236
18	278	282	283	283	**284**	284	284	284	284
20	329	334	**337**	337	337	337	337	337	337
22	385	393	398	398	399	**400**	400	400	400
24	443	452	454	454	**456**	456	456	456	456
26	505	514	517	517	523	**524**	524	524	524
28	566	575	582	582	**589**	589	589	589	589
30	641	655	659	659	661	662	662	**664**	664
32	706	723	726	726	734	**735**	735	735	735
34	780	805	808	808	813	813	814	**815**	815
36	856	878	891	891	903	905	**906**	906	906
38	924	947	955	955	966	971	**972**	972	972
40	1017	1035	1045	1045	1055	1056	1058	**1061**	1061
42	1091	1111	1119	1119	1141	1141	1143	1144	**1146**
44	1175	1205	1207	1207	1225	1229	1230	1230	**1231**
46	1263	1295	1304	1304	1323	1326	1327	1328	**1331**
48	1359	1403	1418	1418	1430	1432	1441	1444	**1446**
50	1447	1489	1497	1497	1512	1519	1522	1522	**1523**

Tab. 4.9: The average performance of ILS based method for different MaxNonImp value when $k = 10$

N	MNI-100	MNI-500	MNI-1000	MNI-2000	MNI-3000	MNI-4000	MNI-6000	MNI-8000	MNI-10000
6	66	**67**	67	67	67	67	67	67	67
8	112	**113**	113	113	113	113	113	113	113
10	168	170	170	**171**	171	171	171	171	171
12	230	232	232	**233**	233	233	233	233	233
14	296	299	**302**	302	302	302	302	302	302
16	374	375	381	382	383	383	**384**	384	384
18	461	464	466	**470**	470	470	470	470	470
20	547	554	559	560	**561**	561	561	561	561
22	648	656	666	669	669	669	**670**	670	670
24	749	757	766	773	775	**776**	776	776	776
26	856	868	882	883	883	884	**885**	885	885
28	973	982	1000	1005	1006	1006	1008	1008	**1010**
30	1105	1114	1126	1131	1133	1136	1136	1136	**1137**
32	1230	1243	1261	1263	1266	1267	**1270**	1270	1270
34	1365	1383	1400	1407	1409	1414	1415	1415	**1416**
36	1515	1530	1555	1562	1564	1567	1570	**1573**	1573
38	1656	1672	1706	1711	1716	1717	1718	**1720**	1720
40	1802	1826	1858	1862	1865	1871	1877	1879	**1880**
42	1965	1993	2026	2033	2039	2041	**2045**	2045	2045
44	2127	2160	2191	2205	2209	2211	2214	**2215**	2215
46	2292	2314	2363	2378	2388	2391	2391	2398	**2405**
48	2475	2512	2552	2556	2569	2573	2580	**2591**	2591
50	2659	2674	2738	2756	2768	2768	2773	2781	**2782**

MaxNonImp values. Note that we reported the average (integer-rounded) results rather than best maximin LHDs in the tables. We observe that though for small values of N MaxNonImp =100, 500 are usually sufficient to obtain the best LHD, these are excessively small values for higher N values, for all k. We also observe that for small values of k, MaxNonImp =1000 is more or less able to obtain the best LHD when $N \leq 25$. But when $N > 25$, improvements are still possible by taking larger MaxNonImp values. Quite expectedly, on average, the suitable MaxNonImp value is increasing with N for all k values.

Figure 4.9 displays the experimental results. In the figure, we plot the N

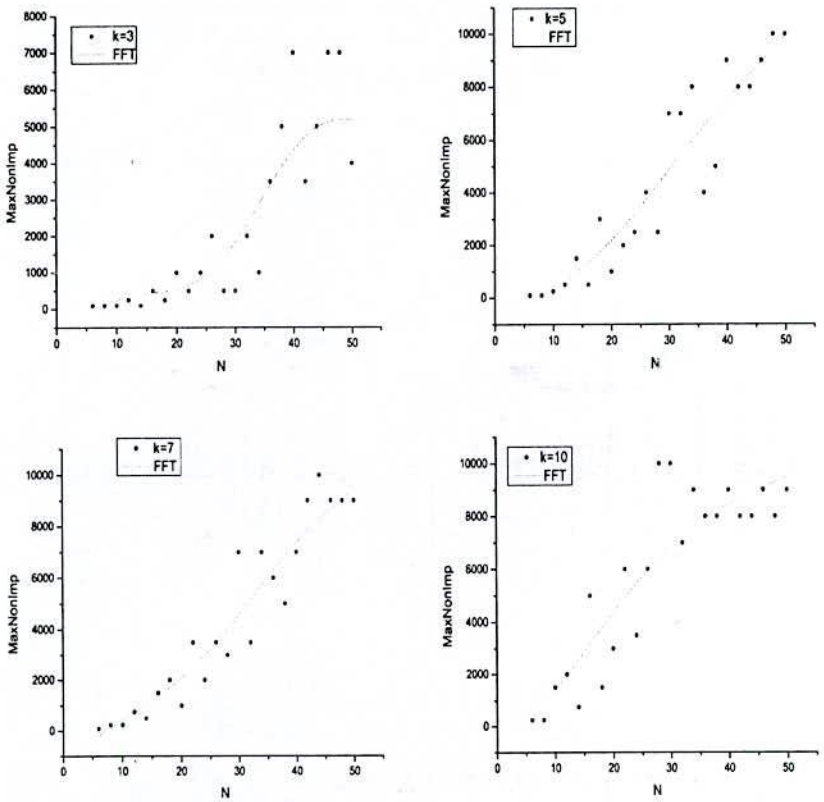

Fig. 4.9: Impact of N on MaxNonImp Parameter for $\text{Mm}(D_1, J_1)$

values on x-axis, versus the "Best-Average" MaxNonImp values (y-axis) for each k value. By the "Best-Average" MaxNonImp, we mean the minimum value of MaxNonImp for which the algorithm obtained the best average maximin distance (Mm) value over 20 runs.

We observe from the figure that the trends of the smoothed curves are quasi-linear for the k values considered. In particular, for $k = 3$, the trend of MaxNonImp is increasing slowly for small values of N and then it is increas-

ing a bit more rapidly. Again for $k = 10$, we observe that the rate of change MaxNonImp is decreasing for large value of N. For these large problems, we conjecture that the phenomenon happens because an even larger MaxNonImp would be needed. We also observe that, on average, the slope of the curves are more or less similar, and linear, for all k, $20 < N < 45$. We can conjecture the existence of a linear relationship relating the suitable MaxNonImp to the N value.

4.3.2 Impact of Dimension on MaxNonImp

For investigating the impact of k on the MaxNonImp parameter, we consider the LHDs: $N = 10, 20, 30, 40$ with all $k = 3, 4, \ldots, 40$. We set `MaxNonImp`= 10,000 as a stopping condition. For each run with `MaxNonImp` =10,000, we store (during the run) the best results for `MaxNonImp`=100,1000, 2500, 2500, 5000, 8000, and 10,000. We considered the average results as usual.

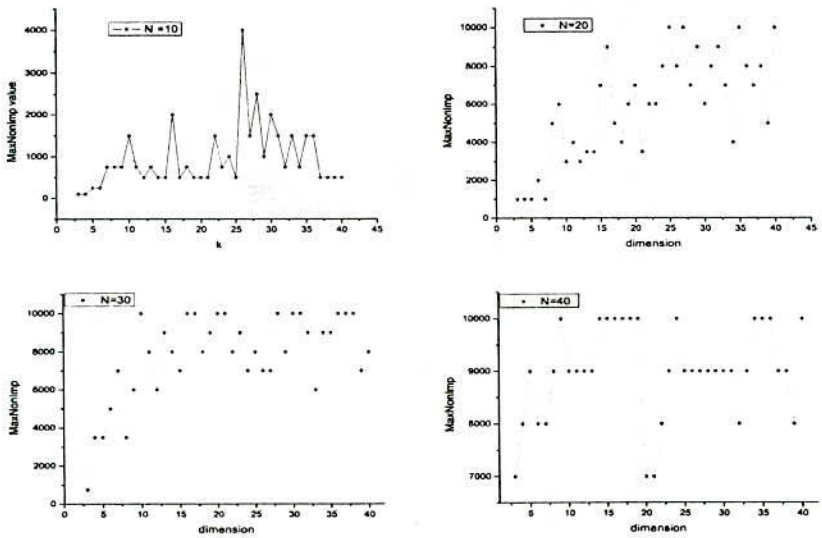

Fig. 4.10: Impact of k on MaxNonImp parameter for $\mathrm{Mm}(D_1, J_1)$

The experimental results are displayed in the Figure 4.10. We observe that when k is small — $k < 11$ — the value of MaxNonImp increases in a roughly logarithmic way with respect to k for $N = 10$ For $N = 20, 30$, the trend is cleaner and, again, somewhat logarithmic. But for $N = 40$ the observed values seem to saturate to MaxNonImp=10000 for $k > 15$, which implies that a larger MaxNonImp would be probably needed.

4.4 Empirical formula for MaxNonImp

Up to now we have observed (in Section 4.3) that the (best observed) MaxNonImp increases in a quasi-linear fashion with respect to N and increases roughly in a logarithmic fashion with k. It is also apparent that a constant MaxNonImp value is not a good strategy for obtaining good maximin LHDs for all N and k values. On the other hand, though a large MaxNonImp value, namely MaxNonImp=10000, is able to obtain better LHDs, of course it is computationally costly. So, now we want to devise an empirical formula (EF) for computing a suitable MaxNonImp value for the given N and k. In what follows we denote the MaxNonImp value produced by EF as MNI-EF.

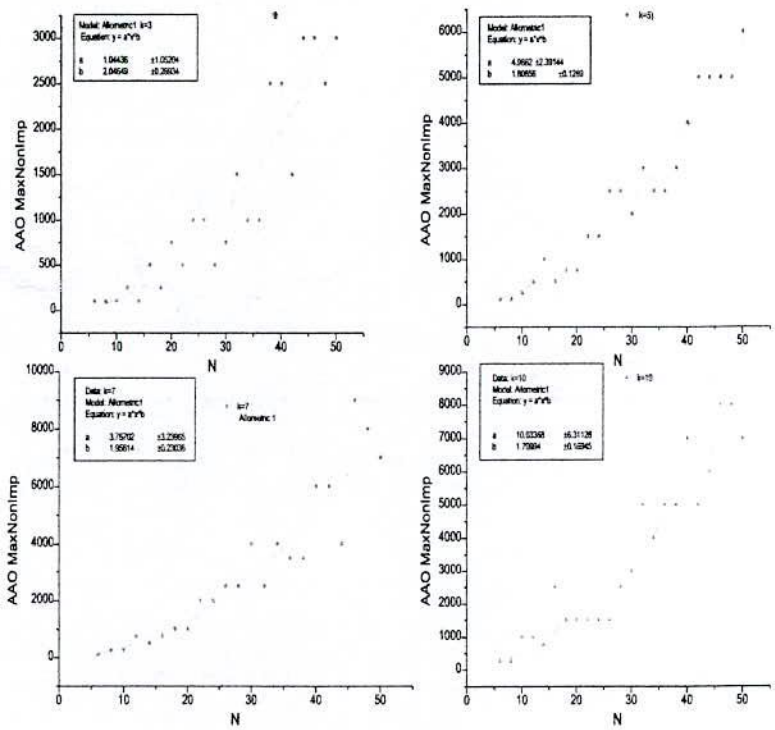

Fig. 4.11: The trend of (a) AA0 MaxNonImp w.r.t. N

We first consider the experiments performed in Section 4.3. We observe in those experiments that sometimes within a large range of MaxNonImp values the average maximin LHDs are unchanged and the change of maximin LHDs value with respect to changes of the MaxNonImp value is small. By considering these circumstances, we display the approximate average optimal (AAO) MaxNonImp

value with respect to N in Figure 4.11. We define the AAO MaxNonImp value as the MaxNonImp value at which a result within a given small percentage of the best one (obviously reached for MaxNonImp=10000) is obtained. In particular, we fixed the percentages 0.4, 0.3, 0.2 and 0.1% respectively for $k = 3, 5, 7, 10$. Just to make the definition clearer, let us consider the following example. For $k = 10$ assume that the best obtained result for MaxNonImp=10000 is 1535. Then, the AAO MaxNonImp value is the first MaxNonImp value at which a result at least equal to $\lfloor 1535 - 0.001 * 1535 \rfloor = 1533$ is reached. Basically, the idea is to consider MaxNonImp values which do not necessarily deliver the best results but for which a further increase of the MaxNonImp value would only slightly improve (at a greater computational cost) the final results.

We notice that the trends are polynomially increasing with respect to N for all k considered. As we also know by the experiments in section 4.3 that there is a logarithmic impact of MaxNonImp on k, we may first assume the empirical formula is of the following form:

$$MaxNonImp = c \times (\log k)^m \times N^n + d, \tag{4.1}$$

where c, d, m and n are constant, N and k correspond to number of design points and number of factors respectively. In order to find out the approximate value of n in equation (4.1), we approximately fit the data for each $k = 3, 5, 7, 10$ as shown in Figure 4.11 (the smooth red lines). We notice that the range of n is $1.70 < n < 2.05$.

To validate the empirical formula (4.1), we first made some primary experiments and then fixed up the values of $a = 0.7$, $b = 50$, $p = 1.90$ and $q = 1.3$ for equation (4.1):

$$MaxNonImp = 0.70 \times (\log k)^{1.3} \times N^{1.90} + 50, \tag{4.2}$$

Figure 4.12 plots the curves of MaxNonImp values generated by formula (4.2) for the same set of problems considered in Figure 4.11: we observe that the two sets of curves are quite similar.

4.4.1 Performance of the empirical formula

In order to investigate the performance of the empirical formula (4.2) we consider LHDs with $k = 3, 5, 7, 10$ and $N = 2i : i = 3, 4, \cdots, 25$. We configured the ILS to use RP local moves with FI local search and SCOE perturbation, and MaxNonImp=10000. $Opt(D_1, J_1)$ is the optimality criterion for all the dimensions and the points. We performed $R = 20$ runs. During the runs we collected the intermediate best LHDs for

- MaxNonImp= 1000,
- the MaxNonImp value computed by the empirical formula (labeled MNI-EF in the tables), and
- MaxNonImp=10000.

We only report the data collected in the experiments with $k = 5$ (displayed in Figure 4.13 and Table 4.10) since those for other k values lead to quite similar

Tab. 4.10: The performance of ILS based method for different MaxNonImp values when $k = 5$

	average LHD			(Opt. LHD, no. of success)			MNI-EF
N	MNI-1000	MNI-10000	MNI-EF	MNI-1000	MNI-10000	MNI-EF	parameter value
6	32	32	32	[32,19]	[32,19]	[32,19]	233
8	48	48	48	[50,1]	[50,1]	[50,1]	294
10	76	76	76	[82,1]	[82,1]	[82,1]	371
12	88	88	88	[90,2]	[90,2]	[90,2]	462
14	110	111	110	[114,2]	[114,2]	[114,2]	569
16	136	136	136	[139,4]	[139,4]	[139,4]	690
18	163	164	163	[168,1]	[168,1]	[168,1]	825
20	191	192	191	[198,1]	[199,1]	[198,1]	975
22	220	222	221	[229,1]	[235,1]	[229,1]	1139
24	247	250	247	[255,1]	[257,2]	[255,1]	1317
26	279	282	280	[288,1]	[288,2]	[288,1]	1508
28	311	314	313	[324,1]	[324,2]	[324,1]	1714
30	348	352	350	[358,1]	[363,1]	[363,1]	1933
32	381	386	385	[394,2]	[402,1]	[402,1]	2166
34	415	425	421	[434,1]	[439,2]	[437,1]	2412
36	457	466	464	[473,1]	[478,1]	[476,1]	2671
38	495	505	501	[512,1]	[526,2]	[526,1]	2944
40	535	547	541	[549,1]	[564,1]	[554,2]	3230
42	576	591	587	[602,1]	[606,1]	[602,2]	3530
44	613	628	627	[638,1]	[652,1]	[652,1]	3842
46	658	673	671	[676,2]	[696,1]	[690,1]	4167
48	707	723	719	[721,2]	[738,1]	[738,1]	4506
50	748	761	759	[768,1]	[781,1]	[781,1]	4857

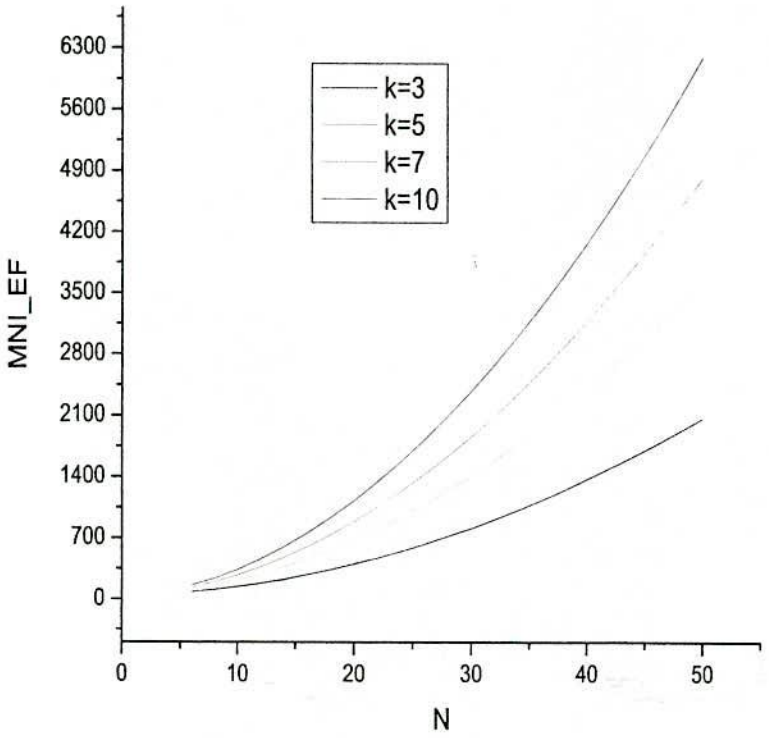

Fig. 4.12: The trend of AA0 MaxNonImp w.r.t. N

conclusions. From Figure 4.13 as well as from Table 4.10, we make two observations. The first one is that when N is relatively small like $N \leq 25$ then the performance of the algorithm shows no significant difference among the three considered MaxNonImp values, for any k values. But when the value of N is large then the algorithm with MNI-EF provides almost always better results compared to the case MaxNonImp=1000. The second observation is that although algorithm with MaxNonImp=10,000 provides better results with respect to that of the algorithm with MaxNonImp=MNI-EF for large values of N, the loss of performance for the MaxNonImp=MNI-EF case is quite limited, while the computational saving is considerable (see Table 4.11).

Now we would like to investigate the computational cost of the above performed experiments. The run times of the experiments are reported in Table 4.11 performed on a machine with a AMD Opteron 1 GHz processor and 4 GB RAM.

As expected, the computational cost of the algorithm with MaxNonImp=MNI-EF is slightly superior with respect to the one with MNI-1000 (the cost for the former is at most twice that of the latter for $k = 10$), whereas the computational

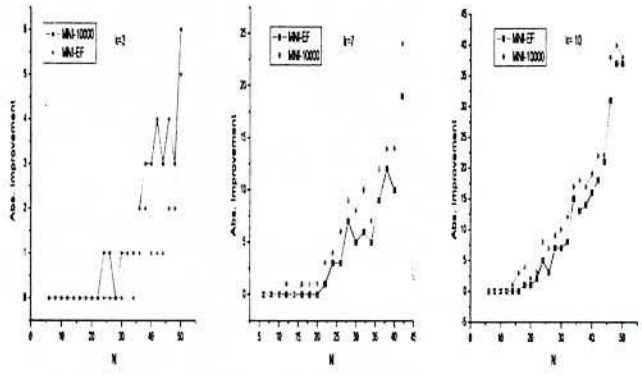

Fig. 4.13: The absolute improvement of MNI-EF and MNI-10000 w.r.t MNI-1000 based approach

Tab. 4.11: Total elapsed time for different `MaxNonImp` values for the ILS approach with the $Opt(D_1, J_1)$ optimality criterion (time is in hours)

k	MNI-10000	MNI-EF	MNI-1000
3	5.6	1.1	0.8
5	20.0	5.7	3.9
7	28.9	9.9	6.1
10	56.2	19.4	9.8
Total T.	110.7	36.1	20.6

cost of the algorithm with MNI-10000 is significantly larger (up to six-seven times larger) than that of the algorithm with MNI-1000.

Given that the gain in the quality with MNI-10000 with respect to MNI-EF is not particularly significative, from these experiments we may claim that the MaxNonImp value produced by formula (4.2) is a quite good compromise between quality of the result and computational cost.

4.5 Comparison of ILS with the existing literature

Now we will compare our results with those in the existing literature. A large number of works about LHDs is available in literature but few of them consider *maximin* LHDs, because the concept of maximin LHD is relatively new. The works on maximin LHDs study general LHDs as well as special types of LHDs like orthogonal LHD, symmetric LHD and so on. For the comparison, here we consider general LHDs; in particular, we compare our results with those reported for the Simulated Annealing (SA) approach in [178] (labeled `SA_M` in the following) and those of the Periodic Design (`PD`) and another Simulated Annealing in what follows denotes as (`SA`) approach presented in the recent paper [117] (also available at the web site `http://www.spacefillingdesigns.nl`).

For ILS, we set RP local moves with BI acceptance rule in local search, SPC

Tab. 4.12: The setting of number of runs for the ILS approach

k	N	R
3-10	2-25	500
3-10	26-50	100
3,4,5	51- 100	50
6-10	51-100	10

perturbation and MaxNonImp=1000. We also consider the $Opt(D_1, J_1)$ optimality criterion. In what follows the approach will be simply denoted as ILS(D_1). For what concerns the number of runs for each LHD, we consider the settings given in Table 4.12.

We report the best maximin *squared* distance for N up to 100 and k ranging from 3 to 10 in Table 4.13.

At first we compare our results with PD and SA reported in Table 4.13. We observe in the table that ILS(D_1) is always at least as good as the SA approach presented in [117] and it is interesting to note that the algorithm got improvements already at low values of N and the improvements tend to become very large as the dimension k increases. Another observation from the table is that ILS(D_1) is also often (much) better than PD but it is dominated for $N > 33, k = 3$ and $N > 95, k = 4$. This is similar to what observed in [117]. In [117] it has been observed that there is a critical number of points above which PD tends to outperform SA. Such critical number quickly increases with the dimension k and, e.g., for $k \geq 6$ such value is certainly above $N = 100$ (note that in [117] results of PD for $k \geq 8$ are not even reported because they are clearly inferior to those of SA for N up to 100). We also remark that ILS(D_1) allows to increase the critical value where PD becomes better than ILS, but for N large enough also ILS(D_1) gets dominated by PD for small k values.

Although the quality of the results obtained by ILS appears to be quite good, it is important to see whether such results have only been obtained by brute force, i.e., at the cost of a very large computational effort. As computational cost is one of the important issue for the heuristic approaches, we report the computational cost of the above experiments in Table 4.14 . The computational cost of PD and SA from [117] is also incorporated in the table. Even taking into account that our CPU is faster than the one of [117], our ILS algorithm is remarkably computationally cheaper. All the batch of tests ran within a week where as Husslage et. al [117] needed more than 4 months to get the results in Table 4.13 for their SA approach.

Now we will compare the results of ILS(D_1) as reported in Table 4.13 with SA_M given in [178]. Note that in [178] only a limited number of such results is available[1]. The comparison is displayed in Table 4.15. Also note that the value

[1] Actually, in [178] the best maximin distance for design points whose components belong to the set $\left\{0, \frac{1}{N-1}, \ldots, 1\right\}$ is reported, but here, with an easy transformation, we report the corresponding maximin squared distance when the components belong to the set $\{0, 1, \ldots, N-1\}$

Tab. 4.13: Comparison between PD, SA and ILS(D_1)

N	$k=3$			$k=4$			$k=5$			$k=6$		
	PD	SA	ILS	PD	SA	ILS	PD	SA	ILS	PD	SA	ILS
2	3	3	3	4	4	4	5	5	5	6	6	6
3	3	6	6	4	7	7	5	8	8	6	12	12
4	6	6	6	12	12	12	11	14	14	15	20	20
5	6	11	11	12	15	15	11	24	24	15	27	27
6	14	14	14	16	22	22	23	32	32	28	40	40
7	14	17	17	16	28	28	23	40	40	28	52	52
8	21	21	21	25	42	42	32	50	50	42	66	66
9	21	22	22	25	42	42	39	61	61	45	76	76
10	21	27	27	36	50	50	55	82	82	62	91	92
11	24	30	30	39	55	55	55	80	81	62	108	110
12	30	36	36	46	63	63	62	91	93	91	136	138
13	35	41	41	51	68	70	64	101	103	91	136	140
14	35	42	42	70	75	78	86	112	115	104	152	156
15	42	48	48	71	83	87	88	124	131	111	167	173
16	42	50	50	85	90	94	101	136	152	130	186	192
17	42	53	54	85	97	100	113	150	158	131	203	210
18	50	56	56	94	103	109	123	162	171	155	223	232
19	57	59	62	94	113	118	136	174	187	169	241	255
20	57	62	66	106	123	130	139	184	201	210	260	279
21	65	66	69	116	127	143	165	201	224	210	283	299
22	69	69	72	117	137	150	174	215	235	223	304	323
23	72	74	77	130	146	158	178	224	247	236	324	343
24	76	78	81	138	154	167	201	242	263	258	343	367
25	91	81	86	156	162	177	205	255	277	286	368	395
26	91	86	89	156	171	186	226	269	288	296	387	411
27	91	90	91	157	178	195	238	287	307	310	410	441
28	94	94	98	174	188	206	258	302	326	339	427	468
29	94	98	101	174	196	214	269	322	343	346	452	490
30	105	102	105	194	209	223	310	335	363	390	473	520
31	107	106	110	212	215	234	310	347	379	390	504	544
32	114	110	116	212	228	246	341	371	396	419	529	572
33	114	113	120	215	234	255	341	379	415	430	548	607
34	133	117	123	230	244	266	358	403	437	470	586	638
35	133	122	129	234	255	278	366	418	462	495	601	657
36	133	129	132	250	261	290	400	427	482	518	631	686
37	152	131	139	266	275	299	408	454	500	528	648	718
38	152	134	142	283	279	309	415	464	525	561	681	755
39	152	139	149	283	290	321	439	486	541	561	706	779
40	155	146	150	291	301	332	492	505	562	632	739	816
41	162	147	155	293	309	348	492	525	590	632	776	851
42	168	152	162	319	325	355	496	543	605	670	791	885
43	168	157	166	323	329	372	520	558	623	670	830	907
44	186	161	174	331	349	378	548	582	653	696	862	943
45	186	166	179	347	362	391	565	615	670	737	891	983
46	186	169	182	366	370	403	592	615	696	797	918	1012
47	186	173	189	378	378	417	611	634	719	797	940	1051
48	189	178	190	413	385	433	632	673	742	857	976	1082
49	196	180	201	415	399	443	634	680	767	893	1015	1119
50	213	185	203	415	414	454	663	699	794	893	1042	1159
51	213	189	206	421	426	469	692	727	804	917	1067	1179
52	213	198	214	455	429	478	709	742	835	1003	1100	1224
53	216	200	217	455	447	493	716	765	858	1003	1136	1252
54	233	213	221	477	454	507	760	783	886	1019	1171	1295
55	243	214	233	483	477	529	760	805	907	1082	1198	1363
56	243	216	233	515	479	537	784	830	939	1104	1236	1384
57	261	221	241	515	490	549	846	854	954	1136	1265	1411
58	261	227	245	539	500	562	846	878	986	1166	1303	1459
59	266	229	248	544	519	574	849	905	1018	1223	1328	1516
60	273	237	254	568	530	591	904	928	1033	1242	1381	1549
61	274	244	259	620	538	609	904	939	1075	1258	1413	1625
62	283	245	266	620	554	621	934	991	1090	1306	1450	1605
63	297	249	275	620	575	630	967	989	1119	1380	1497	1644
64	297	258	278	625	579	645	985	1009	1137	1430	1526	1697
65	314	260	281	630	582	662	997	1035	1170	1430	1565	1760
66	314	269	290	666	602	677	1050	1051	1194	1476	1590	1803
67	314	270	297	666	614	690	1072	1085	1236	1482	1646	1824
68	314	278	297	685	623	707	1087	1119	1243	1538	1664	1899
69	324	280	306	698	650	723	1112	1114	1271	1588	1704	1903
70	325	285	309	716	658	737	1150	1135	1307	1633	1759	1949
71	325	289	317	716	665	750	1150	1187	1326	1644	1783	2014
72	341	296	321	750	678	762	1203	1197	1355	1768	1862	2072
73	350	299	326	759	688	771	1229	1242	1383	1768	1872	2095
74	350	306	332	767	703	794	1229	1269	1426	1774	1910	2151
75	350	310	341	771	714	813	1274	1282	1444	1862	1963	2193
76	363	324	346	813	750	818	1300	1318	1480	1935	2024	2267
77	363	325	350	823	762	849	1308	1331	1508	1947	2051	2298
78	387	337	356	844	761	855	1382	1360	1544	2014	2079	2333
79	387	333	366	848	788	877	1382	1399	1566	2037	2120	2367
80	403	344	366	873	786	898	1395	1430	1606	2037	2152	2400
81	406	338	374	916	782	907	1406	1431	1631	2064	2217	2449
82	406	353	382	938	825	924	1475	1482	1651	2141	2239	2516
83	417	369	387	940	829	936	1501	1509	1695	2141	2290	2562
84	426	363	390	967	838	950	1534	1510	1728	2229	2325	2614
85	426	369	398	967	877	970	1552	1566	1766	2232	2399	2692
86	428	376	403	967	867	985	1573	1578	1783	2375	2437	2732
87	428	374	410	976	877	1017	1598	1589	1811	2375	2476	2715
88	437	374	420	1050	890	1017	1685	1629	1841	2398	2513	2827
89	443	378	422	1050	907	1045	1690	1654	1891	2400	2562	2850
90	481	384	433	1060	940	1062	1710	1696	1909	2516	2633	2927
91	481	393	434	1089	951	1074	1748	1724	1954	2516	2674	2970
92	481	394	446	1089	966	1096	1805	1750	1968	2599	2729	3021
93	481	402	446	1098	962	1110	1813	1795	2006	2604	2726	3073
94	481	405	453	1124	986	1130	1881	1811	2049	2747	2788	3142
95	481	413	465	1135	1010	1141	1901	1846	2066	2747	2817	3169
96	509	414	466	1261	1023	1161	1965	1863	2128	2769	2911	3240
97	515	419	473	1261	1027	1186	1965	1899	2136	2817	2960	3280
98	531	429	477	1261	1055	1196	1965	1929	2170	2850	3001	3354
99	531	449	486	1261	1040	1209	2009	1950	2217	2878	3043	3406
100	554	451	494	1261	1074	1235	2053	1975	2225	3000	3117	3451

N	$k = 7$			$k = 8$		$k = 9$		$k = 10$	
	PD	SA	ILS	SA	ILS	SA	ILS	SA	ILS
2	7	7	7	8	8	9	9	10	10
3	7	13	13	14	14	18	18	19	19
4	16	21	21	26	26	28	28	33	33
5	16	32	32	40	40	43	43	50	50
6	29	47	47	54	54	61	61	68	68
7	31	61	61	70	71	80	81	89	90
8	46	79	80	91	91	101	102	114	114
9	47	93	93	112	113	126	128	141	143
10	68	110	111	130	132	154	157	172	174
11	69	128	131	152	154	178	180	206	209
12	95	150	154	176	181	204	208	235	238
13	95	174	181	202	209	232	240	267	272
14	119	204	215	228	240	265	275	298	309
15	129	211	223	257	276	296	313	337	351
16	155	238	244	286	320	330	354	378	397
17	161	256	267	312	330	367	404	415	445
18	186	281	293	344	360	398	447	458	496
19	195	305	320	370	389	438	470	498	545
20	226	332	348	403	424	472	505	542	607
21	236	361	379	438	460	517	545	592	640
22	270	384	408	467	498	555	587	643	687
23	273	410	444	501	536	596	635	685	733
24	308	444	473	538	576	639	680	739	791
25	350	467	508	583	615	688	732	792	847
26	365	499	535	612	654	726	773	854	899
27	382	526	569	648	699	780	828	896	965
28	406	561	609	693	743	826	879	953	1021
29	417	593	641	733	797	876	942	1015	1090
30	458	620	681	787	838	925	1000	1086	1152
31	482	657	714	812	888	976	1055	1138	1227
32	518	695	757	866	932	1026	1116	1194	1287
33	537	723	792	900	985	1084	1179	1253	1357
34	561	751	828	945	1033	1135	1237	1329	1437
35	586	811	870	1002	1081	1190	1297	1398	1516
36	636	831	919	1042	1135	1257	1359	1459	1584
37	668	863	955	1079	1201	1300	1424	1516	1666
38	709	923	1002	1127	1243	1367	1487	1597	1739
39	726	938	1043	1192	1299	1434	1559	1665	1834
40	786	970	1092	1224	1362	1489	1631	1742	1896
41	802	1016	1133	1271	1421	1562	1707	1820	1994
42	903	1064	1175	1333	1478	1639	1773	1920	2074
43	903	1112	1219	1377	1536	1683	1857	1973	2176
44	903	1140	1268	1463	1589	1752	1922	2072	2249
45	926	1192	1319	1480	1653	1820	1997	2130	2336
46	985	1243	1364	1548	1722	1906	2061	2208	2433
47	985	1268	1418	1616	1775	1958	2161	2331	2534
48	1054	1325	1451	1658	1846	2017	2239	2387	2617
49	1074	1356	1515	1729	1920	2103	2299	2470	2719
50	1113	1397	1549	1772	1985	2179	2404	2556	2808
51	1161	1450	1590	1855	2029	2243	2440	2639	2908
52	1231	1486	1643	1888	2105	2325	2547	2745	2996
53	1241	1537	1700	1949	2162	2429	2617	2825	3079
54	1288	1577	1751	2006	2217	2473	2699	2892	3168
55	1325	1639	1800	2084	2296	2570	2787	3054	3274
56	1358	1701	1849	2162	2368	2623	2913	3100	3407
57	1479	1721	1907	2194	2421	2704	2966	3215	3530
58	1479	1795	1963	2258	2487	2796	3058	3305	3631
59	1509	1821	2028	2356	2564	2881	3134	3399	3700
60	1577	1899	2070	2393	2666	2939	3240	3500	3816
61	1615	1928	2129	2488	2728	3021	3349	3588	3916
62	1680	2023	2182	2541	2813	3132	3432	3700	4069
63	1680	2035	2249	2607	2925	3215	3538	3767	4183
64	1769	2093	2330	2734	2954	3292	3635	3955	4264
65	1786	2132	2364	2723	3059	3357	3718	4034	4410
66	1857	2180	2434	2841	3125	3474	3838	4143	4543
67	1868	2238	2503	2868	3198	3543	3904	4224	4670
68	1940	2295	2581	2956	3266	3647	4031	4360	4756
69	1965	2351	2605	3075	3355	3716	4124	4455	4913
70	2130	2417	2672	3130	3450	3841	4230	4539	5037
71	2130	2451	2732	3161	3531	3936	4366	4689	5153
72	2177	2503	2796	3220	3620	4027	4417	4812	5250
73	2206	2598	2858	3305	3709	4134	4533	4873	5389
74	2244	2614	2909	3432	3807	4224	4658	5038	5521
75	2295	2703	2984	3513	3863	4298	4751	5171	5595
76	2375	2756	3085	3559	3972	4395	4890	5254	5805
77	2403	2819	3134	3617	4044	4492	4989	5399	5932
78	2505	2870	3184	3684	4116	4577	5103	5489	6040
79	2525	2950	3242	3775	4215	4705	5170	5633	6192
80	2590	2979	3324	3877	4290	4807	5283	5773	6361
81	2642	3086	3397	4001	4413	4888	5484	5901	6460
82	2753	3118	3477	3998	4496	5030	5519	6013	6626
83	2767	3195	3534	4076	4558	5102	5672	6097	6747
84	2838	3227	3598	4183	4685	5222	5760	6273	6832
85	2874	3299	3648	4324	4794	5340	5917	6397	7090
86	3103	3335	3824	4397	4889	5423	6038	6491	7189
87	3103	3450	3852	4474	4922	5538	6109	6622	7318
88	3183	3500	3872	4524	5076	5667	6257	6803	7447
89	3183	3541	3940	4578	5140	5774	6361	6872	7668
90	3190	3661	4015	4699	5215	5832	6449	7040	7753
91	3234	3677	4149	4850	5325	5969	6677	7163	7973
92	3277	3760	4228	4873	5404	6081	6741	7286	8128
93	3361	3811	4244	4984	5527	6231	6869	7488	8198
94	3474	3888	4329	5067	5615	6329	6971	7536	8358
95	3531	3940	4408	5154	5754	6396	7137	7741	8523
96	3639	4070	4463	5220	5871	6516	7266	7777	8695
97	3639	4069	4592	5316	5945	6649	7366	8038	8779
98	3690	4147	4644	5445	6111	6776	7542	8242	9070
99	3731	4214	4727	5477	6187	6912	7632	8344	9150
100	3903	4335	4796	5597	6286	6983	7775	8450	9290

Tab. 4.14: Comparison of the elapsed time (in hrs) among PD, SA and ILS(D_1)

k	PD	SA	ILS(D_1)
3	145	500	15
4	61	181	15.53
5	267	152	26.27
6	108	520	11.92
7	232	246	17.49
8	-	460	24.61
9	-	470	39.25
10	-	470	48.24
CPU (M.Hrz)	800 Pen. III	800 Pen. III	1000 AMD Opter.

Tab. 4.15: Comparison between SA_M and ILS(D_1)

	$k=3$		$k=4$		$k=5$		For other value of k & N		
N	SA_M	ILS (D_1)	SA_M	ILS (D_1)	SA_M	ILS (D_1)	(N;k)	SA_M	ILS (D_1)
2	3	3	4	4	5	5	(6; 6)	40	40
3	6	6	7	7	8	8			
4	6	6	12	12	14	14	(7; 7)	61	61
5	11	11	15	15	24	24	(14;7)	**219**	215
6	14	14	22	22	32	32			
7	17	17	28	28	40	40	(8; 8)	91	91
8	21	21	42	42	50	50			
9	22	22	42	42	61	61	(9; 9)	126	**128**
10	27	27	50	50	82	82			
11	29	**30**	55	55	80	**81**			
12	36	36	63(2)	**63(1)**	91	**93**			

within parenthesis in Table 4.15 correspond to J_1 values.

In spite of the fact that in [178] only results with few points are reported, ILS was able to obtain always at least the same quality except for the LHDs $N = 14; k = 7$ but also some improvements. In particular, the improvements are for: (N, k)= $(11, 3), (11, 5), (12, 5), (9, 9)$. We have also an improvement regarding the J_1 value in $(N; k)$= $(12; 4)$. For the single failure $(N; k)$= $(14; 7)$, we mention that a slightly larger number of runs is sufficient to detect the missing best value 219.

In the above experiments we considered PC perturbation based ILS approach. We remark that also Cyclic Order Exchange (COE) allows ILS to outperform SA. And we have already compared the PC perturbation based ILS approach with SA approach in the Table 4.13. In order to make a further comparison between the performance of PC and COE perturbations, we also discuss the results obtained with COE. For this comparison we consider LHDs with $N = 2, \ldots, 100$ and $k = 5$. We set MaxNonImp=1000 and $R = 100$ for all the cases. We display the results in Figure 4.14 in terms of absolute improvement with respect to the SA based approach given in [117]. From this primary experiments, we observe in Figure 4.14 that both ILS approaches outperform

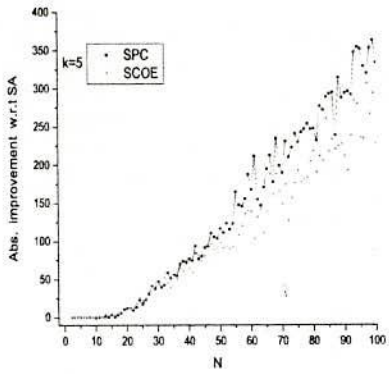

Fig. 4.14: Performance of SPC and SCOE based ILS approaches with respect to SA approach in[178]

SA.

As we discussed earlier (see Section 4.2), for small values of N, the COE based ILS approach works slightly better than the PC perturbation based approach. On the other hand for large value of N, the PC perturbation based ILS approach works better than the COE perturbation based ILS approach.

4.6 *Experiments about the complexity analysis*

The aim of this section is to experimentally assess the computational cost of the proposed ILS(D_1) algorithm. We will first derive the number of operations required by a single local search, and then those for a single run of ILS (from now on in this section we will give as understood that we are discussing the ILS(D_1) version). For these experiments we consider $k = 3, 5, 7, 10$ and $N = 10i : i = 1, 2, \cdots, 10$. We use the following parameter setting : Acceptance criteria= Best Improve(BI), LocalSearch=RP; Stopping criteria: MaxNonImp=1000, Perturbation moves=SCOE and number of trials is one if otherwise not defined.

Assume that we are at iteration s of a local search and that the current value is $D_1^{(s)}, J_1^{(s)}$. The basic operation in a local search is the swap one between two points i and j. In order to compare the new candidate solution with the current one, we need to evaluate $D_1^{(s+1)}$ and $J_1^{(s+1)}$. Such operation does not require to compute from scratch all the distances within the candidate solution. Indeed, only those involving points i and j are changed with respect to the current solution. Therefore, with a proper implementation we should only compute $O(N)$ new distances, each of which requires a number $O(1)$ of operations (indeed, we do not need to compute the distance from scratch but only update the part corresponding to the single coordinate whose value has been changed). In fact we do not always need to compute all the new distances: as soon as we compute a distance lower than $D_1^{(s)}$ we can stop, since the candidate solution is certainly

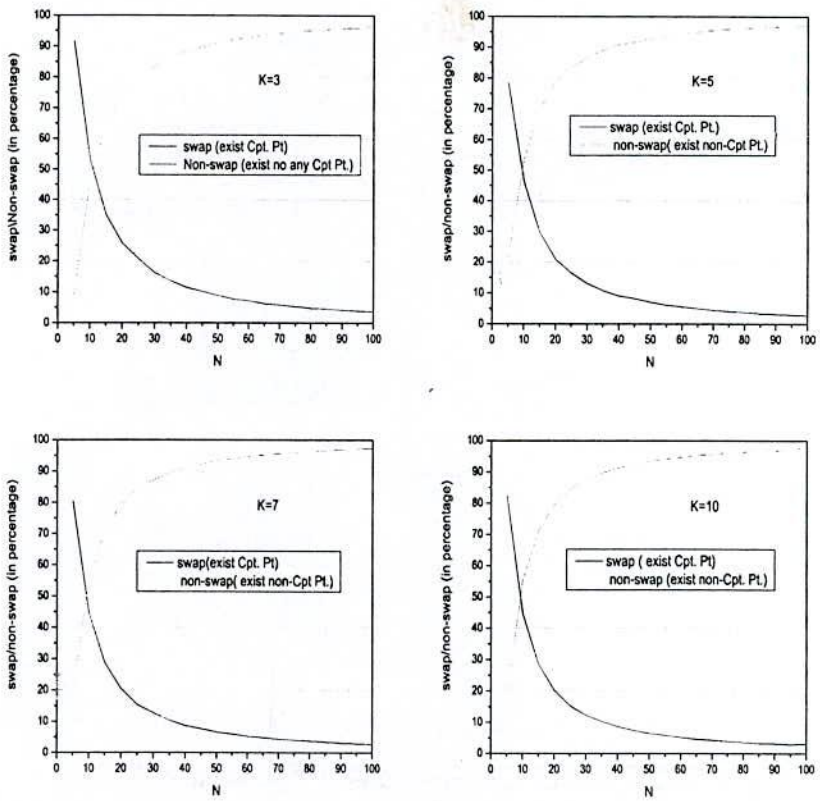

Fig. 4.15: The percentage of pairs involving and not involving critical points

worse than the current one. Therefore, each swap operation requires *at most* $O(N)$ operations.

The number of swap operation is not known in advance. Indeed, swap moves are restricted only to those involving at least a *critical point*. In Figure 4.15 the x-axis reports N and the y-axis reports the percentage of actually analyzed swap moves (those involving at least one critical point) over the total number of possible swaps in each run (those involving all possible pairs of points), for $k = 3, 5, 7, 10$. The black curve represents the percentage of analyzed swaps, the red curve represents the percentage of "avoided" swaps, i.e. those not involving critical points. We observe that for very small N ($N < 14$) most of the possible swaps are to be considered, but as N grows the percentage of swaps to be considered drops dramatically, quickly falling below 10% for $N > 30$.

Figure 4.16 shows the history of the number of critical points during the local searches for the case $(k, N) = (7, 50)$. We observe that most of the times the number of critical points is 1 or 2, and only occasionally is greater than 6. Figure

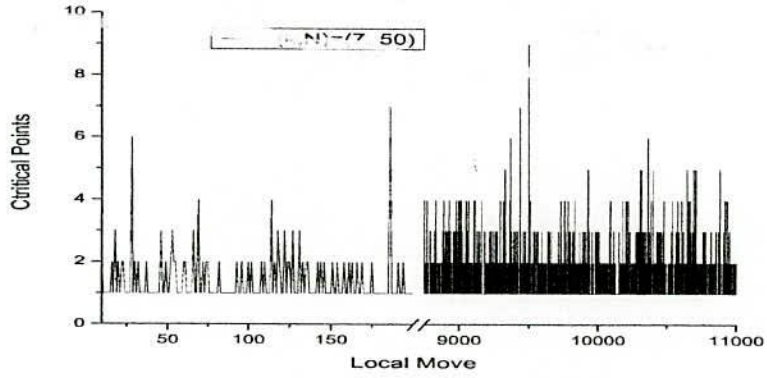

Fig. 4.16: The history of number of critical points for $(k, N) = (7, 50)$ during local move

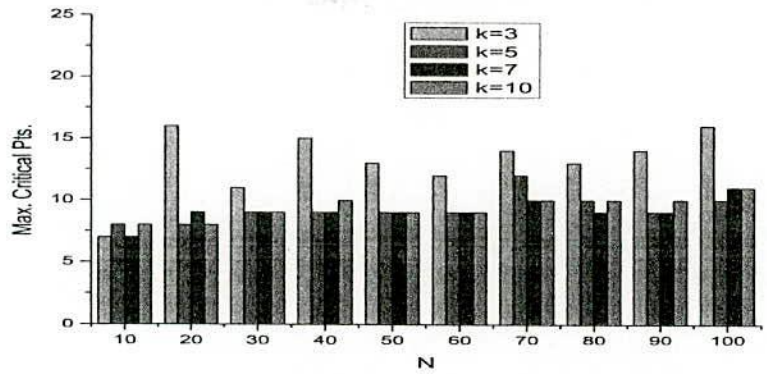

Fig. 4.17: The impact of N on Maximum Critical Points during history of evaluation

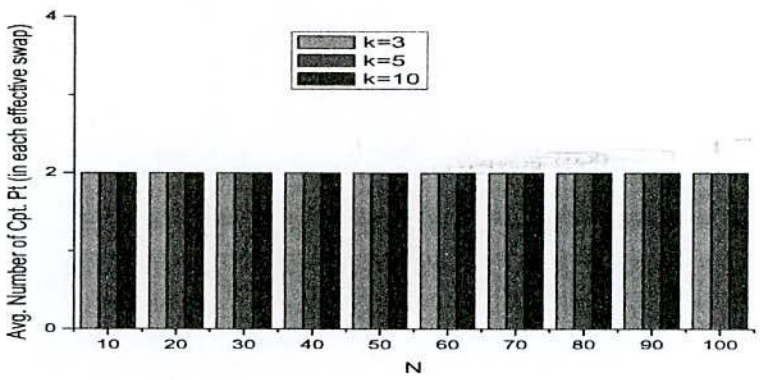

Fig. 4.18: The impact of N on average Critical Points during history of evaluation

4.17 shows with a bar diagram the maximum number of critical points (MCP) obtained during the run of the algorithms for each (k, N). Apparently, we cannot observe any significant impact of N and k on the values of MCP. Indeed such values are always below 20, and most of the time they are near 10. In Figure 4.18 we report the average number of critical points in each neighborhood, rounded to the largest integer; this number is always stuck at 2, for all $k = 3, 5, 10$. So we can confidently claim that the size of the problem has practically no impact on the number of critical points in each visited configuration.
Since we need to consider all the swap moves involving at least one critical point, the above considerations lead us to conclude that the total number of swap moves that we need to perform at a given iteration is simply $O(kN)$ (factor N is due to the number of pairs involving at least a critical point, factor k is due to the fact that, given a pair, we have a swap operation for each possible coordinate).

The last thing we need to consider in order to evaluate the number of operations required by a local search is the number of times the While-Loop is executed, i.e. the number of times an improving solution is observed during a local search. We will denote this number by WL. We will perform some experiments in order to find the impact of N as Figure 4.19 shows the history of WL during LocalSearch for (a) $(k, N) = (7, 20))$ and (b) $(k, N) = (7, 50)$. We observe in Figure 4.19 (a) that most of the time WL lies near 10 and the largest value observed in the figure is less than 25. In Figure (b) we notice that most of the time the number WL lies near 30 and the maximum value of WL (MWL) is near 80. That is WL (average as well as maximum value) increases together with N. Figure 4.20 shows a cleaner representation of the impact of N on the number MWL. Apparently there exists a linear relation between WL and N. We can also observe an impact of the dimension k on WL. In order to investigate the dependence on k, we performed another series of experiments, for $k = 5i : i = 1, 2, \cdots, 10$ and $N = 10, 25, 50, 100$. Note that for finding out the impact on the local search phase, WL is averaged over the corresponding number of performed perturbations. We observe in Figure 4.21 that there is a

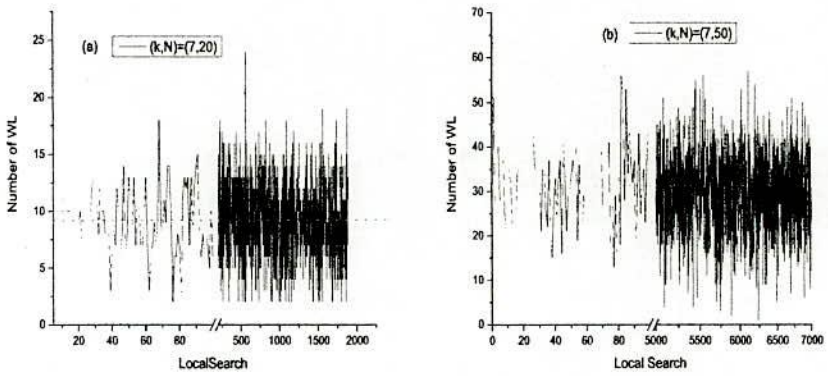

Fig. 4.19: The history of WL for (a)$(k, N) = (7, 20)$; (b)$(k, N) = (7, 50)$ during LocalSearch

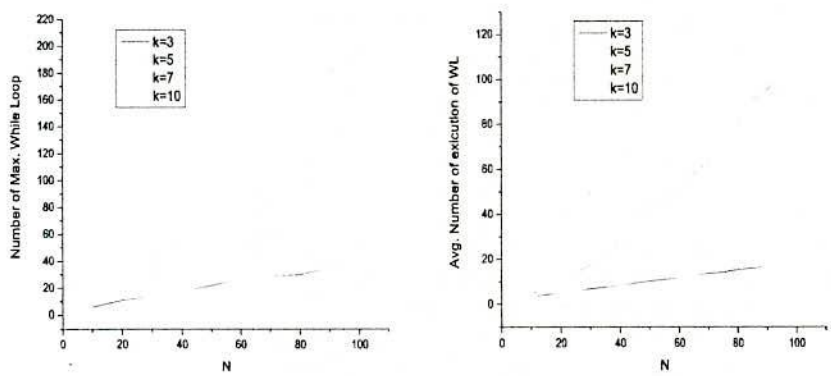

Fig. 4.20: The impact of N on (a) Maximum WL (b) Average WL during history of LocalSearch

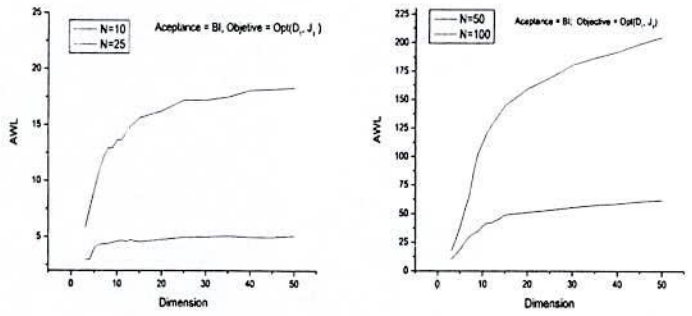

Fig. 4.21: The Impact of k on AWL during LocalSearch

significant impact of k for all $N : N = 10, 25, 50, 100$ on the average number of WL (AWL).

We observe that the trend shown by WL is roughly

$$T \approx k^c$$

where $0 < c < 1$, i.e. a fractional power functional dependence of WL on k. In conclusion, it has been experimentally seen that the number WL is $O(Nk^c)$.

Now, if we sum up the time required by a single swap (at most $O(N)$), the total number of swaps per iteration $O(Nk)$, and the total number of iterations WL $O(Nk^c)$, we conclude that a local search requires at most $O(k^r N^q)$ for some r and q (in particular, we might conjecture that q is close to 3 and r ranges between 1 and 2). In order to validate this result we performed some experiments whose outcome is shown in Figure 4.22. In such figure we report the average computation time per local search as a function of N for the three different values $k = 3, 7, 10$ (the time is the average per local search over 10 runs of ILS). We assume, as derived above, that the approximate time complexity for a local search of the ILS approach is

$$T \approx O(k^r N^q),$$

and will try to determine practical values for r and q experimentally. In Figure 4.22, we observe that, for each k, the curves of elapsed times grow non-linearly with respect to the increasing of N. To find out the approximate value of q we fitted the data in a linear regression for each k as ,

$$\log(T) = q \log(N) + r \log(k),$$

where T = Average elapsed time in each LS. We observe from Figure 4.23 that the range of q is $2.5 < q \leq 3$. In particular, as k increases it seems that q tends to 3, which is the value previously conjectured.

To find out the approximate value of r we performed some experiments by

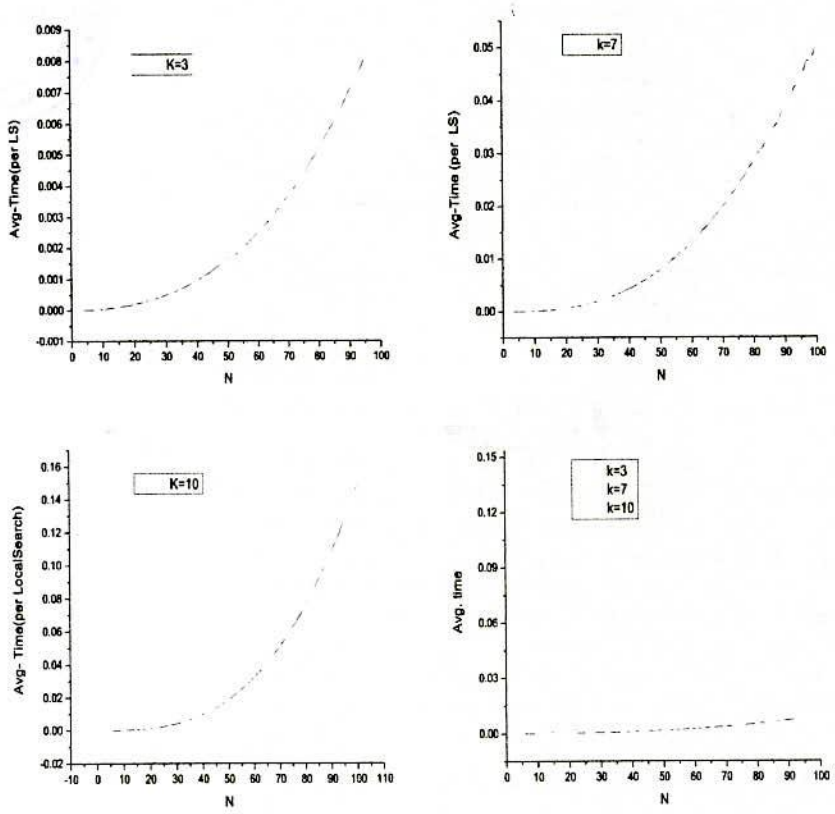

Fig. 4.22: The History of Elapsed time

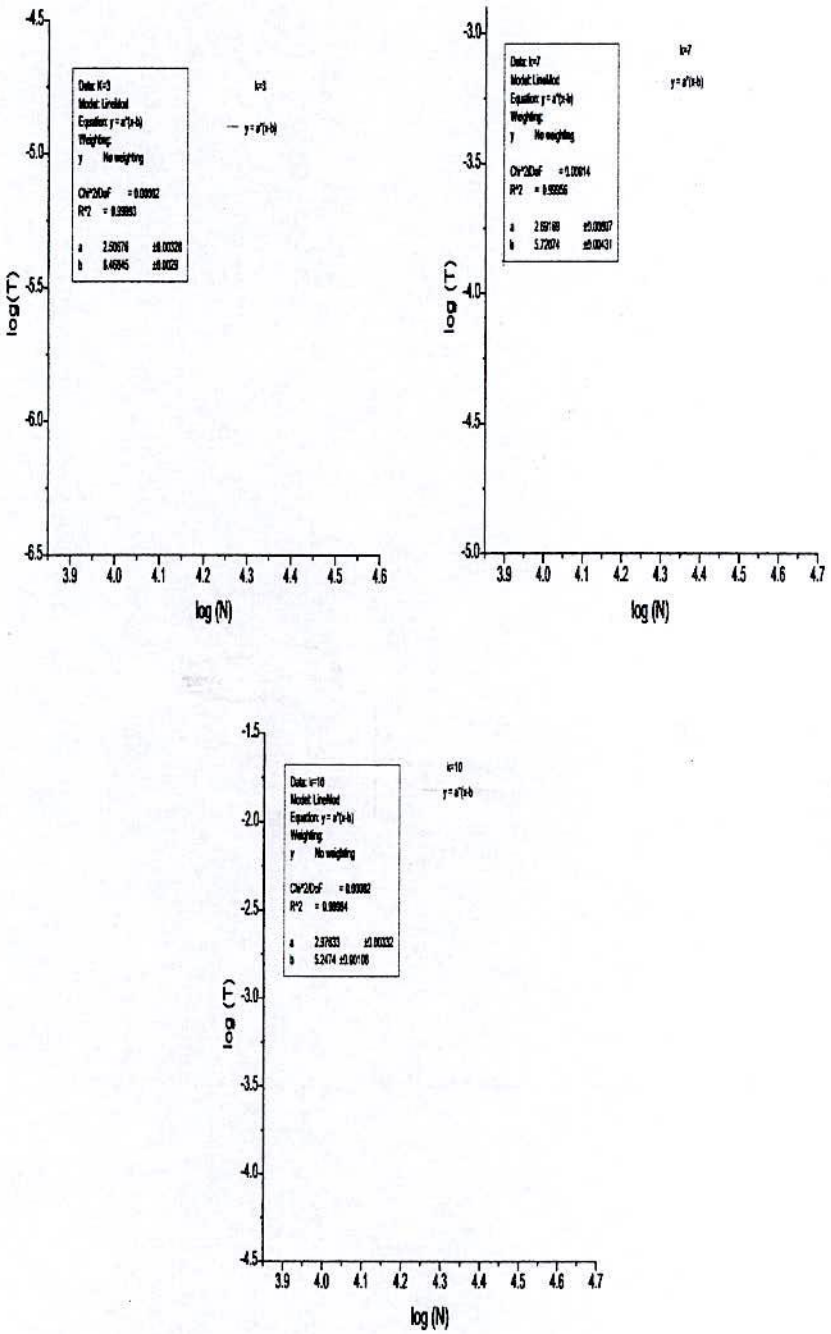

Fig. 4.23: The values of $\log(T)$ plotted with respect to $\log(N)$

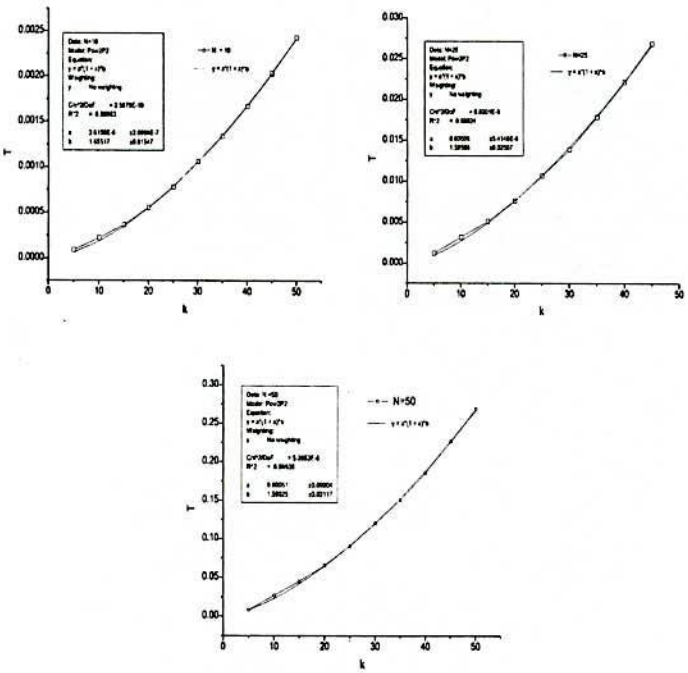

Fig. 4.24: The approximate time complexity for LS with respect to k

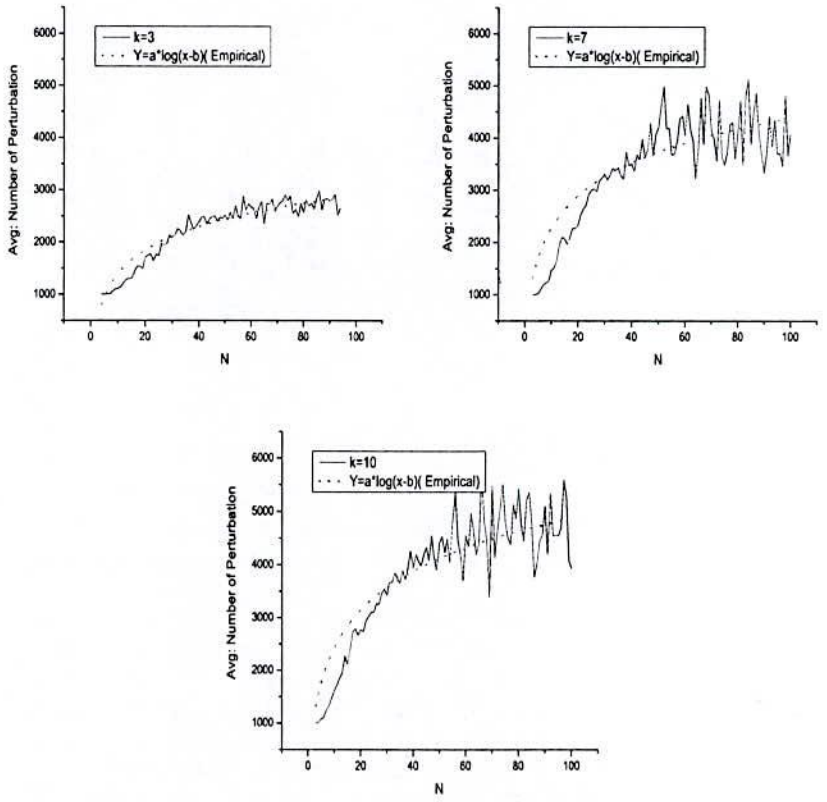

Fig. 4.25: The impact of N on the number of perturbations

considering LHDs $k = 5i : i = 1, 2, \ldots, 10$ with $N = 10, 25, 50, 100$. Figure 4.24 shows the impact of k on average elapsed time in each LS. In the figure we observe that the average elapsed time T increases slightly more than linearly with respect to k. In order to find out the approximate value of r we have fitted the data (see figure 4.24). We notice that the range of r is $1 < r \leq 2$, which is again in accordance with what was previously conjectured.

We remind the reader that this practical time complexity $\approx O(k^r N^q)$, with $r \in (1, 2)$ and $q \in (2, 3)$, for LS has been estimated in the environment of ILS instead of evaluating a stand-alone local search. According to our observations, the local search usually performs less iterations in the ILS environment, due to the "partially optimal" structure preserved by the perturbations.

In order to get to an empirical evaluation of the number of operations required by an ILS run, we still need to evaluate the number of perturbations (and, thus, of local searches) performed during an ILS run. Then, we would like to find out the impact of N as well as k on the number of perturbations during each

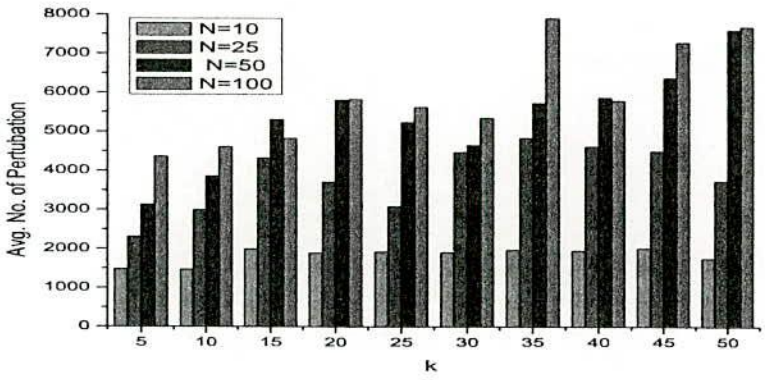

Fig. 4.26: The impact of k on the number of perturbations

ILS run. For these experiments we performed ten runs of ILS and considered the average number of perturbations per run. For these experiments we considered LHDs with $k = 3, 7, 10$ and $N = 3, 4, \ldots, 100$. From the experiments (see Figure 4.25) we notice that there is a significant impact of N on the number of perturbation invoked for all k considered. It seems that the number of invoked perturbations is somewhat logarithmic with respect to N (see the dot curve in Figure 4.25). In order to find the impact of k on the number of perturbations, we considered LHDs: for each $N : N = 10, 25, 50, 100$; $k = 5i : i = 1, \ldots, 10$. From the experiments we remark that there is no significant impact of k on the perturbation invoked during the run (see the bar diagram in Figure 4.26).
Now, if we put together the observation that the overall number of perturbations/local searches per ILS run is $O(\log(N))$, and the previous one about the $O(k^r N^q)$ about the complexity of local searches, we can conclude that a bound on the overall time required by a single ILS run is $O(k^r N^q \log(N))$, a fact that is also experimentally confirmed by the analysis of the elapsed times per ILS run.

5. COMPUTATIONAL EXPERIMENTS AND DISCUSSION ABOUT ILS WITH OPT(ϕ)

In Section 3.2 we proposed two different variants of the ILS approach based on two distinct optimality criteria, namely one corresponding to the pair of values (D_1, J_1) and another corresponding to the previously defined ϕ function. In Section 4 we have already performed experiments on ILS coupled with the optimality criterion based on (D_1, J_1) . In this section we will perform experiments in order to investigate the ILS approach coupled with the optimality criterion based on the ϕ function, and we will compare the relative efficiency of the two ILS approaches.

5.1 Impact of the neighborhood structure and the optimality criterion

In Section 3.2 we have already discussed about the various types of local search procedures. Here we will always adopt Row-wise Pairwise (RP) local moves with FI (First Improve) acceptance rule.

First, we would like to investigate the impact of the neighborhood structure. We will consider RP local moves only involving at least one critical point, denoting them with $\mathcal{LM}_{RpD1}$, and local moves involving also non critical points, denoting them with $\mathcal{LM}_{Rp\phi}$. We will investigate the optimality criterion based on the pair (D_1, J_1), denoted by Opt(D_1, J_1), and the optimality criterion based on function ϕ, denoted as Opt(ϕ). When adopting Opt(ϕ), ILS will return the Mm value (i.e. D_1 value) of its final solution. In fact, a simple but, as we will see, quite meaningful alternative is that of recording the different (D_1, J_1) values observed during a run of ILS and finally returning the best one, which is not necessarily the final one. To distinguish this option from the previous one, we will use a different notation for the optimality criterion, denoted in this case as Opt(D_1, ϕ).

We remark that local moves $\mathcal{LM}_{RpD1}$ and $\mathcal{LM}_{Rp\phi}$ do not deliver different results if the Opt(D_1, J_1) criterion is employed (as already commented, moves not involving at least one critical point can not improve the (D_1, J_1) values), while with the two criteria based on the ϕ function different results can be reached. Note that the neighborhood based on local moves $\mathcal{LM}_{Rp\phi}$ is larger (approximately N times larger) than that based on local moves $\mathcal{LM}_{RpD1}$, so that we should expect larger computational costs when employing the latter local moves.

Experiments are performed with $k = 3$ and $N = 3, 4, \ldots, 25, 5i : i = 6, 7, \ldots, 20$ (the conclusions are quite similar with other k values). We set MaxNonImp=100

Tab. 5.1: Computational experiments with different local moves and optimality criteria for $k = 3$. Note that in the table Opt(D_1, J_1) is denoted as Opt(D).

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
N	Mm Values					ϕ Values				
⇓	Opt(D)	Opt(D_1, ϕ)		Opt(ϕ)		Opt(D)	Opt(D_1, ϕ)		Opt(ϕ)	
nbh	$\mathcal{LM}$	$\mathcal{LM}$	$\mathcal{LM}$	$\mathcal{LM}$	$\mathcal{LM}$	$\mathcal{LM}$	$\mathcal{LM}$	$\mathcal{LM}$	$\mathcal{LM}$	$\mathcal{LM}$
⇒	$RpD1$	$RpD1$	$Rp\phi$	$RpD1$	$Rp\phi$	$RpD1$	$RpD1$	$Rp\phi$	$RpD1$	$Rp\phi$
3	6	6	6	6	6	0.863	0.863	0.863	0.863	0.863
4	6	6	6	6	6	1.227	1.227	1.227	1.227	1.227
5	11	11	11	11	11	1.302	1.302	1.302	1.302	1.302
6	14	14	14	14	14	1.441	1.441	1.441	1.441	1.441
7	17	17	17	17	17	1.594	1.594	1.594	1.591	1.591
8	19	21	21	19	19	1.731	1.739	1.730	1.723	1.723
9	22	22	22	21	21	1.864	1.883	1.890	1.860	1.860
10	27	26	27	26	27	1.935	1.956	1.935	1.951	1.935
11	29	29	29	29	29	2.072	2.049	2.072	2.049	2.049
12	33	33	36	33	36	2.129	2.130	2.101	2.114	2.101
13	41	41	41	41	41	2.168	2.168	2.168	2.137	2.137
14	41	41	41	41	41	2.284	2.309	2.304	2.282	2.280
15	48	43	45	42	43	2.311	2.399	2.368	2.392	2.359
16	46	49	50	49	49	2.461	2.459	2.447	2.459	2.428
17	51	53	51	53	51	2.567	2.553	2.540	2.553	2.534
18	56	54	54	54	54	2.624	2.647	2.631	2.647	2.598
19	56	57	59	56	56	2.741	2.730	2.683	2.720	2.664
20	61	59	62	59	62	2.792	2.800	2.753	2.800	2.742
21	66	65	66	65	66	2.863	2.862	2.822	2.862	2.814
22	69	68	74	66	72	2.911	2.932	2.856	2.927	2.849
23	73	73	75	73	75	2.993	2.989	2.939	2.989	2.939
24	77	76	77	76	77	3.044	3.053	3.007	3.053	3.007
25	81	78	84	78	81	3.114	3.108	3.072	3.108	3.058
30	98	98	105	98	104	3.443	3.411	3.335	3.410	3.303
35	121	120	129	118	129	3.677	3.672	3.553	3.666	3.553
40	146	145	150	142	146	3.870	3.895	3.781	3.894	3.764
45	166	164	179	164	179	4.094	4.077	3.968	4.077	3.968
50	187	186	204	163	181	4.318	4.318	4.162	4.032	4.110
55	214	213	227	185	211	4.502	4.494	4.349	3.791	4.040
60	238	238	258	209	234	4.679	4.655	4.506	4.176	4.397
65	262	260	286	229	281	4.856	4.855	4.651	3.898	4.634
70	294	292	309	254	288	4.986	4.998	4.809	4.312	4.709
75	309	314	345	280	318	5.190	5.153	4.923	4.758	4.921
80	342	341	371	300	341	5.315	5.297	5.084	4.229	5.077
85	371	370	406	298	370	5.447	5.419	5.209	4.595	5.044
90	398	401	437	384	389	5.582	5.506	5.324	5.276	5.007
95	421	429	474	387	440	5.731	5.688	5.452	4.409	5.258
100	454	458	494	389	473	5.851	5.826	5.577	4.245	5.174
Tot. Time (hrs)		Opt(D);$\mathcal{LM}_{RpD1}$ = 0.24		Opt(D_1, ϕ);$\mathcal{LM}_{RpD1}$ = 10.14			Opt(D_1, ϕ);$\mathcal{LM}_{Rp\phi}$ =15.90			

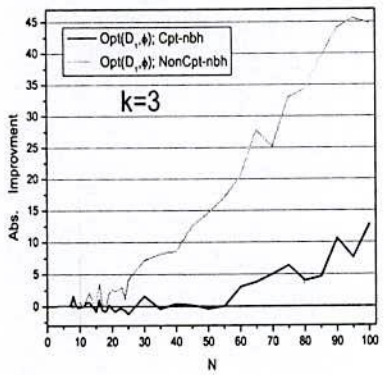

Fig. 5.1: The absolute improvement (regarding average Mm values) of ILS with optimality criterion Opt(D_1, ϕ) and local moves $\mathcal{LM}_{RpD_1}$ (dark curve) and $\mathcal{LM}_{Rp\phi}$ (light curve) with respect to ILS with the Opt(D_1, J_1) optimality criterion.

when using the more time consuming local moves $\mathcal{LM}_{Rp\phi}$, and set MaxNonImp=1000 when employing $\mathcal{LM}_{RpD1}$ local moves. For all the cases we used the SCOE perturbation moves and we considered a number $R = 5$ of ILS runs. The value of p, the parameter of ϕ, has been fixed to 20 for all the experiments.
All the results of the experiments are displayed in Table 5.1 and Figures 5.1 and 5.2. Note that in the figures the names Cpt-nbh and NonCpt-nbh correspond to neighborhoods based respectively on $\mathcal{LM}_{RpD_1}$ and $\mathcal{LM}_{Rp\phi}$ local moves.

Table 5.1 contains the optimal Maximin (Mm) values and the corresponding ϕ values obtained with the two different local moves $\mathcal{LM}_{RpD_1}$ and $\mathcal{LM}_{Rp\phi}$ and the three different optimality criteria Opt(D_1, J_1), Opt(D_1, ϕ) and Opt(ϕ). Figure 5.1 displays the trend of the absolute improvement regarding *average* (over the 5 runs) Mm values obtained with the Opt(D_1, ϕ) optimality criterion and local moves $\mathcal{LM}_{RpD_1}$ (dark curve) and $\mathcal{LM}_{RpD_1}$ (light curve), with respect to the corresponding values obtained by the Opt(D_1, J_1) optimality criterion. Figure 5.2 displays the history of the average number of local moves per local search with different local moves and/or optimality criteria.

In the following subsections we would like to discuss the outcomes of these experiments.

5.1.1 A comparison between the Opt(D_1, ϕ) and Opt(ϕ) optimality criteria

We know that both when employing the Opt(D_1, ϕ) and when employing the Opt(ϕ) optimality criterion, the search is driven by the ϕ function which we aim at minimizing. The only difference between the two strategies is that the latter returns the (D_1, J_1) values of the final solution returned by ILS, while the former returns the best (D_1, J_1) values observed during an ILS run. The cost of the two approaches is basically the same (with Opt(D_1, ϕ) we only have

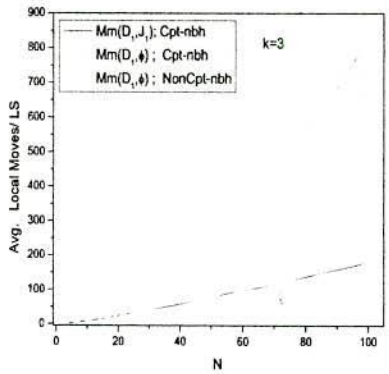

Fig. 5.2: Average number of local moves history among the three ILS approaches – (a) $\mathcal{LM}_{RpD_1}$ neighborhood structure with Opt(D_1, J_1) optimal criterion, (b) $\mathcal{LM}_{RpD_1}$ neighborhood structure with Opt(D_1, ϕ) optimal criterion and (c) $\mathcal{LM}_{Rp\phi}$ neighborhood structure with Opt(D_1, ϕ) optimal criterion

the additional mild cost of keeping track of the best values observed during a run). Moreover, it is guaranteed that the results with Opt(D_1, ϕ) are always at least as good as those with Opt(ϕ). This suggests that it is always advisable to employ Opt(D_1, ϕ). Here, however, we would like to give a measure of the improvements of Opt(D_1, ϕ) with respect to Opt(ϕ). In Table 5.1 columns C3, C4 report the best observed Mm values (with corresponding ϕ values given in columns C8 and C9 respectively) obtained with Opt(D_1, ϕ), while columns C5, C6 report the corresponding Mm values for Opt(ϕ) (the corresponding ϕ values, which are also the best ones observed during the ILS runs, are given in columns C10 and C11 respectively). As expected, we observe that for both the neighborhood structures considered here (the one based on $\mathcal{LM}_{RpD_1}$ local moves and the one based on $\mathcal{LM}_{Rp\phi}$ local moves), on based approaches, the Mm values obtained by Opt(D_1, ϕ) are often better than those obtained by Opt(ϕ), but, even more important, we observe that the improvement becomes more consistent as we increase the number N of points. We also point out that, according to other experiments not reported here, further improvements are observed when also the dimension k is increased.

By comparing the Mm values with the corresponding ϕ values, it becomes clear what we previously already commented, i.e. that a monotonic search with respect to ϕ is not necessarily a monotonic one with respect to (D_1, J_1), so that while performing a monotonic search with respect to ϕ we are in fact performing a search through the (D_1, J_1) values in which some sort of controlled backtracking (acceptance of worsening moves) is present. For instance, we see in Table 5.1 that for the case $(N, k) = (70, 3)$, the minimum ϕ_{20} values is 4.709 with corresponding Mm value =288 obtained with Opt(ϕ). On the other hand, with Opt(D_1, ϕ) we obtain a better Mm value = 309, while the corresponding ϕ value is worse, namely equal to 4.809 (recall that the ϕ value has to be minimized). This is just an example but looking at columns C8-C11 many other similar

examples can be found.

5.1.2 Comparison of Opt(D_1, J_1) and Opt(D_1, ϕ) with $\mathcal{LM}_{RpD1}$ local moves

We have observed that it is always worthwhile to employ the Opt(D_1, ϕ) rather than the Opt(ϕ) one. Next, we compare Opt(D_1, ϕ) with Opt(D_1, J_1). At first we will make the comparison when the local moves employed are only those involving at least one critical point, denoted by $\mathcal{LM}_{RpD1}$.

We observe in the table (consider the columns C2 and C3) that the Mm values obtained by the ILS approaches with the two different optimality criteria are comparable when we consider $\mathcal{LM}_{RpD1}$ local moves. In Figure 5.1 we notice that for small N values the quality of the average Mm values of the two approaches is basically the same, while as N increases the quality with Opt(D_1, ϕ) gets slightly better. On the other hand, the computational costs of Opt(D_1, J_1) are significantly lower than those with Opt(D_1, ϕ) (see the last row of the table). The considerably larger times with Opt(D_1, ϕ) and local move $\mathcal{LM}_{RpD1}$ with respect to those with Opt(D_1, J_1), were somehow not expected. A possible explanation is that, while the neighborhoods explored at each iteration by the local search are of similar size, the number of improvements detected during a local search (i.e. the number of times the While-Loop is executed during a local search) is much larger when the local search is driven by the objective function ϕ rather than (D_1, J_1). Seen in another way, we can say that the search landscape based on the (D_1, J_1) values is flatter (many configurations may have the same (D_1, J_1) values but, at the same time different ϕ values). This way it is easier for the search based on the (D_1, J_1) values to get stuck at some solution without being able to make any progress, and the algorithm stops after fewer iterations than the one where the search is driven by the ϕ values.

5.1.3 Comparison of Opt(D_1, J_1) with $\mathcal{LM}_{RpD1}$ local moves and Opt(D_1, ϕ) with $\mathcal{LM}_{Rp\phi}$ local moves

Here we will compare the results obtained by Opt(D_1, J_1) and $\mathcal{LM}_{RpD1}$ local moves (see column C2), with those obtained by Opt(D_1, ϕ) and $\mathcal{LM}_{Rp\phi}$ local moves (see column C4). We notice that the latter approach is able to obtain significatively better Mm values compared to the former one. This can also be seen in Figure 5.1, where we considered the average Mm values returned by the $R = 5$ ILS runs rather than the best observed Mm value. On the other hand the computational cost of Opt(D_1, J_1) is considerably lower than that of Opt(D_1, ϕ). Then, we might conjecture that the higher quality is only due to the larger computational times. Of course, this is one reason but not the only one. To check this we made a further experiment: we increased the number of ILS runs with Opt(D_1, J_1) in such a way that the overall computational time with the two optimality criteria was similar, and then we compared the Mm values. It turns out that even with this larger number of runs ILS with Opt(D_1, J_1) was unable to reach the same quality of ILS with Opt(D_1, ϕ). So which is the reason for the improved quality of the results obtained by Opt(D_1, ϕ) with $\mathcal{LM}_{Rp\phi}$ local moves? It seems to us that there are two main causes:

- the search space of the neighborhood based on $\mathcal{LM}_{Rp\phi}$ local moves is obviously larger than that based on $\mathcal{LM}_{RpD1}$ local moves (approximately

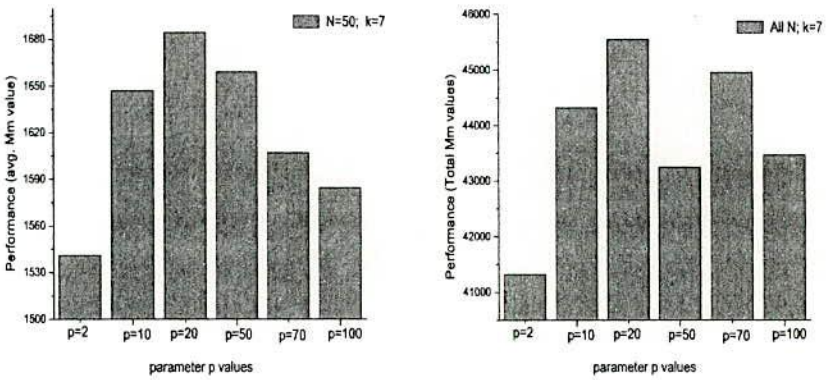

Fig. 5.3: The comparison of different p values

N times larger due to the inclusion of swap moves not involving critical points);

- when performing a monotonic search with respect to ϕ we are in fact performing a non monotonic one with respect to the (D_1, J_1) values, i.e., as already commented, we have some form of backtracking in the search, which prevents the algorithm from being too greedy.

In particular, we point out that the second cause was not really expected before starting the experiments and we believe that the importance of this form of backtracking controlled by the ϕ values is a quite interesting observation.

5.2 Impact of parameter p

In view of the previously obtained results, here we will only consider $\mathcal{LM}_{Rp\phi}$ local moves with the Opt(D_1, ϕ) optimality criterion. We would like to discuss more thoroughly the choice of the parameter p appearing in the definition of the ϕ function, which up to now has been simply fixed to 20. Recall that for large enough p, each term in the sum of ϕ (see (3.4 in Section 3) dominates all subsequent terms and as $p \to \infty$, the optimality criterion based on ϕ becomes equivalent to that based on (D_1, J_1). A practical issue is to establish a "good" value for p. Such value should not be too small, because it would not differentiate enough between the different D_i, J_i values, but at the same time it should not be too large, because too large a value would lead to the search based on the (D_1, J_1) values, which, as previously observed, is too rigid, too greedy. In order to investigate the impact of parameter p, we considered ILS based on the ϕ function with the values $p = 2, 10, 20, 50, 70, 100$. For all the p values, we set MaxNonImp=100 with SCOE perturbation moves and number of runs $R = 5$. We considered $N = 3, 4 \ldots, 25, 5i : i = 6, 7 \ldots, 20$ and $k = 7$. The experimental results are reported in Table 5.2 and are also displayed in Figure 5.3

We notice in Table 5.2 that $p = 20$ is able to obtain very often better Mm values, while $p = 50$ and $p = 70$ are able to obtain better Mm values only few

Tab. 5.2: Computational experiments with different p values

N	$p=2$	$p=10$	$p=20$	$p=50$	$p=70$	$p=100$
3	13	13	13	13	13	13
4	21	21	21	21	21	21
5	32	32	32	32	32	32
6	47	47	47	47	47	46
7	61	61	61	60	61	60
8	79	79	**80**	79	79	78
9	88	93	**94**	93	92	92
10	105	110	**111**	110	110	108
11	127	129	**132**	132	132	128
12	154	155	**155**	153	152	152
13	181	181	**182**	182	179	178
14	216	216	**218**	216	216	215
15	207	219	**224**	**224**	**224**	223
16	238	246	**248**	**248**	240	241
17	251	266	**270**	**270**	267	266
18	277	295	295	295	**296**	284
19	300	322	**327**	324	324	320
20	333	363	**355**	352	353	345
21	366	382	**385**	384	382	381
22	396	414	**421**	417	414	410
23	435	453	455	**457**	451	449
24	472	489	**495**	491	476	481
25	513	518	**527**	526	522	511
30	666	712	**716**	**716**	697	694
35	879	919	**936**	924	900	900
40	1053	1116	1141	**1149**	1128	1122
45	1277	1360	**1400**	1389	1376	1302
50	1561	1658	**1696**	1663	1643	1632
55	1886	1978	**2006**	**2043**	1961	1870
60	2030	2230	**2277**	2273	2240	2124
65	2331	2482	2547	**2579**	2542	2421
70	2586	2783	2856	**2895**	2861	2757
75	2886	3130	**3230**	2895	3225	3141
80	3262	3472	3586	3219	**3597**	3333
85	3571	3862	3984	3641	**3997**	3804
90	3943	4223	4362	4023	**4400**	4223
95	4279	4617	4784	4394	**4844**	4613
100	4664	4997	**5206**	4856	5099	5036
Tot. Best value (40)	05	05	29	10	11	03
Tot. time (hrs)	122	294	324	186	148	60

times. Very small p values (2,10), and very large ones (100), are unable to get the best results except at very small N values. If we have a look at the average Mm values (see Figure 5.3), then we notice that the extreme values $p = 2$ and $p = 100$ still give poor results, while $p = 10$ delivers results of reasonable quality, although not the best ones. Even in this comparison the superiority of $p = 20$ is quite evident. Less evident is the reason why at $p = 50$ we have results which are not particularly satisfying. More investigations will be needed to clarify this fact. From these experiments we infer that the ϕ function is not effective in driving the search when either too many (small p) or excessively few (large p) distance values D_i play a major role in determining the ϕ value. The best option seems to be an intermediate p value and, in particular, $p = 20$ seems to be a robust choice (though not necessarily always the best one as we also observed in the experiments).

Regarding the computational costs, we notice in the table that the higher quality obtained with $p = 20$ has somehow to be paid. Indeed, this choice needs more CPU time than all the other ones. We note that the cost tends to decrease with the p value. This is probably due to the fact that, apparently, increasing p causes a faster convergence to locally optimal solutions and a reduction of the exploration of the search space.

5.3 Impact of MaxNonImp parameter

We have already investigated the impact of the MaxNonImp parameter in our proposed ILS approach for the Opt(D_1, J_1) optimality criterion. Now we perform some experiments to investigate the impact of MaxNonImp in the proposed ILS approach with the Opt(D_1, ϕ) optimality criterion. For these experiments we consider $\mathcal{LM}_{Rp\phi}$ local moves with FI acceptance rule, and SCOE perturbation moves; $N = 2i : i = 3, 4, \ldots, 25$ and $k = 3, 7, 10$. Tests are performed with the following values MaxNonImp= 50, 100, 250, 500, 750, 1000, 1250, 1500, 1750 and 2000. We perform $R = 5$ runs of ILS for each LHD. The experimental results (average Mm values) are displayed in Tables 5.3-5.5, while the best observed Mm values and the corresponding number of times such values are attained out of 5 runs are reported in Table 5.6 (data in the square bracket –the first term indicates the best Mm value and the second term indicates number of times it has been attained).

Though a bit irregular, the results in the tables show that:

- a good value of MaxNonImp tends to increase with N and k (as largely expected);
- with respect to Opt(D_1, J_1) it seems that appropriate MaxNonImp values are considerably smaller (around ten times smaller); indeed, for instance, the best Mm values at large k and N values tend to stabilize very often at MaxNonImp=1000 or even less, while with Opt(D_1, J_1) stability was often reached close to MaxNonImp=10000. This fact allows to partially counterbalance the much higher computational times required by the search based on the Opt(D_1, ϕ) optimality criterion.

As we have previously done with the Opt(D_1, J_1) optimality criterion, we could derive also here an Empirical Formula (EF) giving an appropriate value for

Tab. 5.3: Average results with different MaxNonImp values for $k = 3$

N	MNI-50	MNI-100	MNI-250	MNI-500	MNI-750	MNI-1000	MNI-1250	MNI-1500	MNI-1750	MNI-2000
6	**14**	14	14	14	14	14	14	14	14	14
8	**19**	19	19	19	19	19	19	19	19	19
10	26	26	**27**	27	27	27	27	27	27	27
12	33	33	33	**34**	34	34	34	34	34	34
14	**41**	41	41	41	41	41	41	41	41	41
16	48	**49**	49	49	49	49	49	49	49	49
18	**54**	54	54	54	54	54	54	54	54	54
20	**62**	62	62	62	62	62	62	62	62	62
22	69	69	69	**70**	70	70	70	70	70	70
24	76	78	79	80	**81**	81	81	81	81	81
26	86	86	86	**88**	88	88	88	88	88	88
28	94	95	96	97	97	97	**98**	98	98	98
30	102	103	105	**107**	107	107	107	107	107	107
32	111	113	114	**116**	116	116	116	116	116	116
34	121	122	124	125	125	125	125	125	**126**	126
36	128	129	132	132	133	133	133	**134**	134	134
38	138	139	141	144	144	**145**	145	145	145	145
40	148	148	151	**153**	153	153	153	153	153	153
42	157	160	162	163	**165**	165	165	165	165	165
44	168	170	174	175	175	175	**176**	176	176	176
46	181	182	**183**	183	183	183	183	183	183	183
48	188	190	195	196	**197**	197	197	197	197	197
50	199	201	204	**205**	205	205	205	205	205	205

Tab. 5.4: Average results with different MaxNonImp values for $k = 7$

N	MNI-50	MNI-100	MNI-250	MNI-500	MNI-750	MNI-1000	MNI-1250	MNI-1500	MNI-1750	MNI-2000
6	**45**	45	45	45	45	45	45	45	45	45
8	78	78	**79**	79	79	79	79	79	79	79
10	108	109	**110**	110	110	110	110	110	110	110
12	152	**154**	154	154	154	154	154	154	154	154
14	215	216	**217**	217	217	217	217	217	217	217
16	246	247	247	248	248	248	**249**	249	249	249
18	292	295	296	296	**297**	297	297	297	297	297
20	353	358	359	359	359	359	359	359	**360**	360
22	414	417	419	420	421	**422**	422	422	422	422
24	488	489	492	492	**493**	493	493	493	493	493
26	555	556	559	561	**562**	562	562	562	562	562
28	626	630	634	634	**637**	637	637	637	637	637
30	703	706	711	713	**714**	714	714	714	714	714
32	789	793	796	796	797	**799**	799	799	799	799
34	882	884	887	889	889	889	889	889	**890**	890
36	957	962	968	971	971	971	971	971	971	**972**
38	1040	1042	1044	1045	1048	1048	1048	**1049**	1049	1049
40	1138	1140	1145	1147	1150	1150	**1151**	1151	1151	1151
42	1233	1234	1236	1238	1239	**1241**	1241	1241	1241	1241
44	1332	1337	1340	1342	1342	1342	1343	1343	**1344**	1344
46	1440	1446	1449	1449	1452	1452	**1453**	1453	1453	1453
48	1547	1560	1569	1570	1571	1572	1572	1572	**1574**	1574
50	1684	1690	1694	1695	1697	1697	**1699**	1699	1699	1699

Tab. 5.5: Average results with different MaxNonImp values for $k = 10$

N	MNI-50	MNI-100	MNI-250	MNI-500	MNI-750	MNI-1000	MNI-1250	MNI-1500	MNI-1750	MNI-2000
6	**67**	67	67	67	67	67	67	67	67	67
8	113	**114**	114	114	114	114	114	114	114	114
10	172	173	**174**	174	174	174	174	174	174	174
12	238	238	239	**240**	240	240	240	240	240	240
14	311	313	313	**314**	314	314	314	314	314	314
16	402	403	404	404	405	**406**	406	406	406	406
18	505	507	508	**509**	509	509	509	509	509	509
20	639	639	641	643	643	643	**644**	644	644	644
22	698	699	700	702	702	703	703	703	703	**704**
24	801	807	808	808	**809**	809	809	809	809	809
26	922	927	930	933	933	**934**	934	934	934	934
28	1053	1057	1060	1061	1061	1062	1062	**1063**	1063	1063
30	1197	1198	1198	1199	1199	1200	1200	1200	**1201**	1201
32	1341	1347	1349	1350	1351	1351	1351	1351	1351	**1352**
34	1500	1502	1507	1511	1511	1511	1511	1511	**1512**	1512
36	1670	1672	1675	1675	**1676**	1676	1676	1676	1676	1676
38	1845	1848	1855	1855	**1857**	1857	1857	1857	1857	1857
40	2067	2071	2077	2078	2078	2079	**2080**	2080	2080	2080
42	2174	2178	2182	2189	2189	**2190**	2190	2190	2190	2190
44	2350	2353	2358	2363	2363	2363	2363	2363	**2364**	2364
46	2546	2549	2557	2558	2561	2562	2563	2563	2563	**2564**
48	2742	2750	2757	2763	2763	2764	2764	2767	2767	**2768**
50	2970	2974	2978	2980	**2981**	2981	2981	2981	2981	2981

Tab. 5.6: Best Mm values and number of times they are attained over $R = 5$ runs with different MaxNonImp values

N	$k = 3$			$k = 7$			$k = 10$		
	MNI-100	MNI-1000	MNI-2000	MNI-100	MNI-1000	MNI-2000	MNI-100	MNI-1000	MNI-2000
6	[14,5]	[14,5]	[14,5]	[46,1]	[47,1]	[47,1]	[67,4]	[67,5]	[67,5]
8	[19,5]	[21,1]	[21,1]	[79,1]	[79,3]	[79,3]	[114,4]	[114,5]	[114,5]
10	[27,1]	[27,4]	[27,4]	[110,1]	[111,1]	[111,1]	[174,1]	[174,5]	[174,5]
12	[33,5]	[36,1]	[36,1]	[155,2]	[155,3]	[155,3]	[240,1]	[241,1]	[241,1]
14	[42,1]	[42,2]	[42,2]	[218,1]	[218,1]	[218,1]	[314,2]	[315,1]	[315,2]
16	[50,1]	[50,2]	[50,2]	[250,1]	[250,1]	[250,2]	[405,1]	[407,2]	[407,2]
18	[54,5]	[56,1]	[56,1]	[297,1]	[300,2]	[300,2]	[509,1]	[512,1]	[512,1]
20	[62,5]	[62,5]	[62,5]	[363,1]	[364,1]	[365,1]	[642,1]	[645,1]	[645,1]
22	[70,1]	[73,1]	[73,1]	[419,2]	[425,1]	[425,1]	[701,1]	[704,2]	[704,3]
24	[81,1]	[81,5]	[81,5]	[490,2]	[496,1]	[496,1]	[812,1]	[813,1]	[813,1]
26	[86,5]	[89,2]	[89,2]	[562,1]	[565,1]	[565,1]	[932,1]	[946,1]	[946,1]
28	[98,1]	[98,3]	[98,3]	[635,1]	[641,1]	[641,1]	[1070,1]	[1070,1]	[1070,1]
30	[105,1]	[108,1]	[108,1]	[712,1]	[725,1]	[725,1]	[1204,1]	[1207,1]	[1210,1]
32	[116,1]	[117,1]	[117,1]	[795,1]	[800,3]	[800,3]	[1353,1]	[1354,1]	[1354,1]
34	[123,1]	[126,2]	[126,3]	[890,1]	[897,1]	[897,1]	[1506,1]	[1514,1]	[1514,1]
36	[131,1]	[134,3]	[134,5]	[965,1]	[978,1]	[978,1]	[1675,1]	[1690,1]	[1690,1]
38	[140,1]	[146,4]	[146,4]	[1047,1]	[1053,1]	[1055,1]	[1867,1]	[1870,1]	[1870,1]
40	[150,1]	[154,3]	[154,3]	[1145,1]	[1162,1]	[1162,1]	[2088,1]	[2102,1]	[2102,1]
42	[162,1]	[166,1]	[168,1]	[1237,2]	[1249,1]	[1249,1]	[2186,1]	[2198,1]	[2198,1]
44	[173,1]	[178,1]	[182,1]	[1347,1]	[1347,2]	[1347,2]	[2369,1]	[2373,1]	[2374,1]
46	[184,1]	[186,1]	[186,1]	[1461,1]	[1470,1]	[1470,1]	[2564,1]	[2570,1]	[2573,2]
48	[195,1]	[200,1]	[201,1]	[1569,1]	[1586,1]	[1586,1]	[2759,1]	[2774,1]	[2774,1]
50	[203,1]	[210,1]	[210,1]	[1697,1]	[1707,1]	[1707,1]	[2983,1]	[2986,1]	[2986,1]
Time (hrs)	1.73	16.14	25.96	17.51	116.21	190.17	28.83	197.81	325.31

MaxNonImp as a function of N and k. However, in what follows we will not look for the best compromise between quality of the results and computation times, but only perform experiments with MaxNonImp=100, which will turn out to be already quite significative.

5.4 Further experiments

In this section we compare the performance of ILS coupled with the Opt(D_1, ϕ) optimality criterion with the existing literature as well as with the previously discussed ILS approach coupled with the Opt(D_1, J_1) optimality criterion (whose results are already reported in Section 4.5). The experiments about ILS with Opt(D_1, ϕ) have been performed with the following settings:- Local search procedure: local move $\mathcal{LM}_{Rp\phi}$ with FI acceptance rule; perturbation moves : SCOE; MaxNonImp=100; $p = 20$; LHDs : $N = 2, 3, \ldots, 100$ with $k = 3, 4, \ldots, 10$. The number of runs for each (N, k) pair are reported in Table 5.2. In the same table we also recall the number of runs performed with Opt(D_1, J_1). Note that due to the larger computational times required by Opt(D_1, ϕ) with $\mathcal{LM}_{Rp\phi}$ local moves, we performed considerably fewer runs with this setting with respect to ILS with Opt(D_1, J_1).

5.4.1 Comparison of ILS approaches based on the Opt(D_1, J_1) and Opt(D_1, ϕ) optimality criteria

At first we compare the performance of ILS coupled with the Opt(D_1, ϕ) optimality criterion (in what follows simply denoted as ILS(ϕ)) with that of ILS coupled with the Opt(D_1, J_1) optimality criterion (in what follows simply denoted as ILS(D_1) again). The experimental results are displayed in Figure 5.4. In Figure 5.4(a) we report the absolute improvement (regarding Mm values) of ILS(ϕ) with respect to ILS(D_1) for each N and $k = 4, 7, 10$. Figure 5.4(b) displays the relative increase in the elapsed time of ILS(ϕ) with respect to ILS(D_1). We notice that for $N < 20$ the performance of the two approaches are compara-

Tab. 5.7: The number R of ILS runs (with the optimality criteria Opt(D_1, J_1) and Opt(D_1, ϕ)) for the different (N, k) pairs

k	(a) Opt(D_1, J_1)			(b) Opt(D_1, ϕ)		
	$N =$ 2-25	$N =$ 26-50	$N =$ 51-100	$N =$ 2-25	$N =$ 26-50	$N =$ 51-100
3	500	100	50	100	10	10
4	500	100	50	100	10	10
5	500	100	50	10	10	5
6	500	100	10	10	10	5
7	500	100	10	10	10	5
8	500	100	10	10	10	5
9	500	100	10	10	10	2
10	500	100	10	10	10	2

ble for all the k values considered both from the point of view of the quality and from the point of view of the computation times. But for $N > 20$, we notice in Figure 5.4(a) that the absolute improvement of ILS(ϕ) increases quite quickly both with N and with k. In Figure 5.4(b) we notice that, in spite of the lower number of runs, ILS(ϕ) is still more computationally demanding with respect to ILS(D_1). The figure does not show a regular pattern due to the changes in the number of runs which occur at different N and k values. However, we can notice that there are minor differences up to $N = 50$, while for $N > 50$ the computation times for ILS(ϕ) never gets larger than twice those of ILS(D_1). However, it is worthwhile to recall, that according to some further experiments over a limited set of instances, increasing the computational cost of ILS(D_1) at the same or even at a higher level with respect to that of ILS(ϕ) by increasing the number of runs, was not enough to reach the same Mm values obtained by ILS(ϕ) (see Table 5.8). We emphasize once again that the superiority of ILS(ϕ), mostly motivated by the non monotonic search through the D_1 values induced by the monotonic search through the ϕ values, appears to us as a quite remarkable fact. The searches performed by ILS(ϕ) are definitely more costly, but, according to the results, such larger costs are well paid in terms of quality of the results.

5.4.2 Comparison of ILS(ϕ) with the existing literature

Now we would like to compare our experimental results with those available in literature as well as the latest results available at the web site [276]. For the comparison with the existing literature, we will refer to the same approaches already considered in Section 4.5, namely the approach in [178], denoted as SA_M, and the approaches proposed in [117], denoted as PD and SA approaches. We will denote the updated web site values as `Web` (or `BestKnown`) values. These are improvements obtained through the PD and SA approaches discussed in [117]. The last update was done in January, 21st 2008. Note that, differently from the results reported in the paper [117], the computation times to deliver the updates are not reported in the web site.

At first we compare our approach with the `SA_M` approach. The comparison

Tab. 5.8: Results for ILS(D_1) and ILS(ϕ) with comparable computation times (in seconds) over some instances

(N,k)	Mm Values		elapsed times	
	ILS(D_1))	ILS((ϕ)	ILS(D_1))	ILS(ϕ)
(13, 5)	103	104	91.2	35.5
(14, 5)	115	116	113.1	60.1
(17, 5)	158	159	201.3	78.6
(18, 5)	171	172	240.2	160.3
(19, 5)	187	189	315.1	255.4
(20, 5)	201	206	375.8	312.0
(21, 5)	224	229	438.4	307.3
(24, 5)	263	269	690.1	452.9
(25, 5)	277	286	781.9	737.4
(11, 7)	131	132	96.8	65.4
12(, 7)	154	155	126.7	94.7
(14, 7)	215	217	200.3	135.7
(25, 7)	508	531	1450.2	1269.1
(12 , 10)	238	240	239.9	111.0
(13 , 10)	272	275	309.5	191.9
(14 , 10)	309	313	389.8	275.1
(15 , 10)	351	358	492.5	418.8
(16, 10)	397	406	623.0	600.1
(17, 10)	445	458	757.9	800.2
(18, 10)	496	509	927.9	942.5
(19, 10)	545	569	1185.9	834.4
(20 , 10)	607	641	1441.4	692.3
(21, 10)	640	648	1697.9	588.1
(22, 10)	687	704	2031.0	1738.9
(23 , 10)	733	750	2269.8	1881.3
(24, 10)	791	818	2668.7	2925.3
(25, 10)	847	875	3146.1	1914.0

Tab. 5.9: Comparison between SA_M and ILS(ϕ)

N	$k=3$		$k=4$		$k=5$		For others k & N		
	SA_M	ILS (ϕ)	SA_M	ILS (ϕ)	SA_M	ILS (ϕ)	(N;k)	SA_M	ILS (ϕ)
2	3	3	4	4	5	5	(6; 6)	40	40
3	6	6	7	7	8	8			
4	6	6	12	12	14	14	(7; 7)	61	61
5	11	11	15	15	24	24	(14;7)	**219**	217
6	14	14	22	22	32	32			
7	17	17	28	28	40	40	(8; 8)	91	91
8	21	21	42	42	50	50			
9	22	22	42	42	61	61	(9; 9)	126	***127***
10	27	27	50	50	82	82			
11	29	**30**	55	55	80	**81**			
12	36	36	63(2)	**63(1)**	91	**93**			

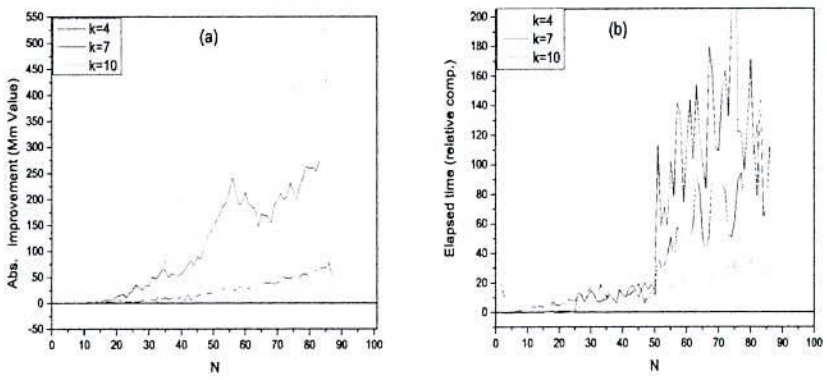

Fig. 5.4: Comparison between the performance of ILS(D_1) and ILS(ϕ)

results are given in Table 5.9. We observe that there are five improvements obtained by ILS(ϕ) and a failure (the same as ILs(D_1)). It is also worthwhile to remark that ILS(ϕ) is even able to improve the solution for the case $(N, k) = (14, 7)$ (with a Mm value equal to 220, see Appendix A) and to get the same result as ILS(D_1) in the case $(N, k) = (9, 9)$ by increasing the number of runs and/or changing different p values.

Now we compare our approach with PD and SA approaches presented in [117] as well as with the updated results reported in [276]. The comparison results are displayed in Table 5.10. We observe that ILS(ϕ) is able to detect a very large amount of improved solutions with respect to the best known ones. This is in particular true at large k values. For $k \geq 6$, with the exception of few number of low N values, all the solutions returned by ILS(ϕ) are improvements of the best known results.

It is interesting to have a closer look at the comparison with the PD approach. Although this usually delivers worse results, as already previously discussed, it seems that the PD approach tends to perform better and better as N increases. Figure 5.5(a) displays the absolute improvement and Figure 5.5(b) displays the relative improvement of the ILS approach with respect to the PD approach for $k = 3, 4$. Figure 5.6 displays the same information for $k = 5, 6, 7$. The improved performance of PD as N increases can be clearly noticed in Table 5.10 and in Figure 5.5 for $k = 3, 4$. But it can also be remarked in Figure 5.6(d) where we notice that the relative improvement of ILS(ϕ) with respect to PD tends to decrease as N increases (although the absolute improvement is clearly increasing).

We still need to comment about the computation times. As already remarked we do not have information about those required to obtain the `Web` results. It is however quite clear that ILS(ϕ) is more computationally demanding with respect to PD and SA. Such higher costs are clearly rewarded in terms of quality of the results but we might wonder about the quality of the results if we impose time restrictions on ILS. According to some further experiments that we per-

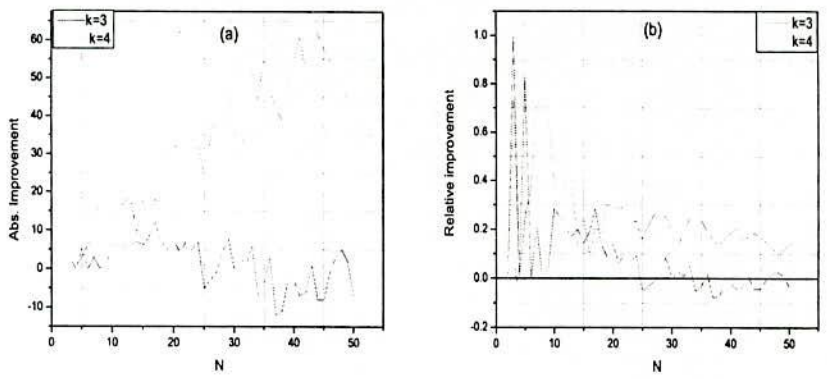

Fig. 5.5: Comparison between PD and ILS(ϕ) for $k = 3, 4$

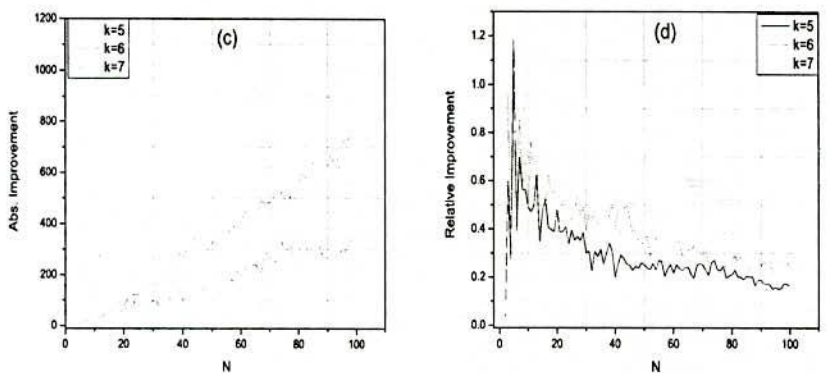

Fig. 5.6: Comparison between PD and ILS(ϕ) for $k = 5, 6, 7$

formed, we could realize that, especially at large k values, equivalent or better results with respect to the PD and SA ones, could quickly be reached by ILS(ϕ). Therefore, it seems that at large k values even few and short runs of ILS(ϕ) are able to deliver results better than those reached by PD and SA.

5.5 *Experiments about the complexity analysis of ILS(ϕ)*

In this section we perform some experiments to derive a formula connecting the computation times of ILS(ϕ) with N and k. The analysis will be similar to the one previously done for ILS(D_1). For these experiments we consider ILS(ϕ) with the following setting: Local Search: acceptance criterion=First Improve (FI), local move=$\mathcal{LM}_{Rp\phi}$; stopping criterion: MaxNonImp=100; Perturbation Technique=SCOE.

Tab. 5.10: Comparison between PD, SA, Web and ILS(ϕ) results. Note that Times are in hours.

	$k=3$				$k=4$				$k=5$				$k=6$			
N	PD	SA	Web	ILS (ϕ)	PD	SA	Web	ILS (ϕ)	PD	SA	Web	ILS (ϕ)	PD	SA	Web	ILS (ϕ)
2	3	3	3	3	4	4	4	4	5	5	5	5	6	6	6	6
3	3	6	6	6	4	7	7	7	5	8	8	8	6	12	12	12
4	6	6	6	6	12	12	12	12	11	14	14	14	15	20	20	20
5	6	11	11	11	12	15	15	15	11	24	24	24	15	27	27	27
6	14	14	14	14	16	22	22	22	23	32	32	32	28	40	40	40
7	14	17	17	17	16	28	28	28	23	40	40	40	28	52	52	52
8	21	21	21	21	25	42	42	42	32	50	50	50	42	66	66	66
9	21	22	22	22	25	42	42	42	39	61	61	61	45	76	76	76
10	21	27	27	27	36	50	50	50	55	82	82	82	62	91	91	**92**
11	24	30	30	30	39	55	55	55	55	80	80	**81**	62	108	108	**110**
12	30	36	36	36	46	63	63	63	62	91	91	**93**	91	136	139	139
13	35	41	41	41	51	68	70	70	64	101	103	**104**	91	136	138	**140**
14	35	42	42	42	70	75	77	**79**	86	112	114	**116**	104	152	154	**160**
15	42	48	48	48	71	83	87	**89**	88	124	129	**131**	111	167	171	**175**
16	42	50	50	50	85	90	93	**94**	101	136	151	**154**	130	186	190	**194**
17	42	53	53	**54**	85	97	99	**103**	113	150	158	**159**	131	203	208	**214**
18	50	56	56	**57**	94	103	108	**111**	123	162	170	**172**	155	223	231	**241**
19	57	59	59	**62**	94	113	119	**122**	136	174	184	**189**	169	241	256	**263**
20	57	62	65	**66**	106	123	130	**137**	139	184	206	206	210	260	279	**285**
21	65	66	68	**69**	116	127	145	**149**	165	201	223	**229**	210	283	302	**306**
22	69	69	72	**76**	117	137	150	**151**	174	215	235	**242**	223	304	325	**338**
23	72	74	75	**77**	130	146	159	**161**	178	224	250	**251**	236	324	348	**358**
24	76	78	81	**83**	138	154	170	170	201	242	266	**269**	258	343	374	**378**
25	91	81	91	86	156	162	178	**181**	205	255	285	**286**	286	368	400	**408**
26	91	86	91	89	156	171	188	**189**	226	269	302	**306**	296	387	426	**439**
27	91	90	91	91	157	178	198	198	238	287	310	**326**	310	410	447	**474**
28	94	94	94	**98**	174	188	210	**212**	258	302	331	**349**	339	427	479	**494**
29	94	98	101	**102**	174	196	221	219	269	322	349	**373**	346	452	507	**517**
30	105	102	105	105	194	209	233	230	310	335	367	**403**	390	473	531	**545**
31	107	106	110	110	212	215	244	240	310	347	405	**406**	390	504	563	**569**
32	114	110	114	**116**	212	228	253	252	341	371	413	**418**	419	529	587	**599**
33	114	113	117	**120**	215	234	264	**267**	341	379	426	**446**	430	548	622	**634**
34	133	117	133	126	230	244	273	**274**	358	403	445	**460**	470	586	648	**668**
35	133	122	133	129	234	255	286	**289**	366	418	467	**482**	495	601	683	**697**
36	133	129	133	**136**	250	261	297	**298**	400	427	486	**502**	518	631	719	**739**
37	152	131	152	140	266	275	309	308	408	454	520	**530**	528	648	744	**775**
38	152	134	152	142	283	279	321	**322**	415	464	541	**557**	561	681	788	**813**
39	152	139	152	149	283	290	330	330	439	486	566	**575**	561	706	816	**846**
40	155	146	155	152	291	301	342	**345**	492	505	575	**590**	632	739	876	**886**
41	162	147	162	155	293	309	355	354	492	525	596	**618**	632	776	882	**938**
42	168	152	168	162	319	325	367	**371**	496	543	626	**641**	670	791	907	**988**
43	168	157	171	169	323	329	383	378	520	558	666	664	670	830	947	**996**
44	186	161	186	178	331	349	396	393	548	582	680	**688**	696	862	992	**1041**
45	186	166	186	179	347	362	407	405	565	615	698	**706**	737	891	996	**1065**
46	186	169	189	185	366	370	421	421	592	615	723	**728**	797	918	1064	**1107**
47	186	173	189	189	378	378	438	426	611	634	754	**762**	797	940	1088	**1113**
48	189	178	201	194	413	385	450	**451**	632	673	774	**782**	857	976	1119	**1159**
49	196	180	203	201	415	399	464	463	634	680	803	799	893	1015	1167	**1181**
50	213	185	213	206	415	414	478	473	663	699	830	**830**	893	1042	1203	**1218**
51	213	189	213	209	421	426	490	487	692	727	850	**857**	917	1067	1230	**1258**
52	213	198	217	214	455	429	504	501	709	742	883	874	1003	1100	1274	**1292**
53	216	200	219	**221**	455	447	515	**516**	716	765	894	**901**	1003	1136	1340	**1340**
54	233	213	233	227	477	454	534	526	760	783	932	**935**	1019	1171	1359	**1392**
55	243	214	243	233	483	477	546	541	760	805	956	**966**	1082	1198	1421	**1432**
56	243	216	243	235	515	479	558	**565**	784	830	982	**992**	1104	1236	1431	**1484**
57	261	221	261	241	515	490	574	570	846	854	1007	**1018**	1136	1265	1488	**1523**
58	261	227	261	246	539	500	594	591	846	878	1035	**1046**	1166	1303	1554	**1559**
59	266	229	266	254	544	519	609	607	849	905	1063	**1064**	1223	1328	1564	**1615**
60	273	237	273	258	568	530	618	**622**	904	928	1094	**1101**	1242	1381	1631	**1647**
61	274	244	274	262	620	538	630	**641**	904	939	1128	**1134**	1258	1413	1667	**1703**
62	283	245	283	269	620	554	657	645	934	991	1150	**1156**	1306	1450	1715	**1756**
63	297	249	297	276	620	575	670	666	967	989	1178	**1187**	1380	1497	1781	**1781**
64	297	258	297	281	625	579	684	678	985	1009	1206	**1223**	1430	1526	1804	**1834**
65	314	260	314	286	630	582	694	**701**	997	1035	1216	**1239**	1430	1565	1868	**1884**
66	314	269	314	294	666	602	718	706	1050	1051	1261	**1272**	1476	1590	1874	**1926**
67	314	270	314	297	666	614	733	726	1072	1085	1299	1283	1482	1646	1954	**1977**
68	314	278	314	306	685	623	746	738	1087	1119	1330	**1360**	1538	1664	1983	**2014**
69	324	280	324	310	698	650	765	754	1112	1114	1351	**1399**	1588	1704	2028	**2070**
70	325	285	325	313	716	658	779	773	1150	1135	1378	**1439**	1633	1759	2094	**2116**
71	325	289	325	325	716	665	793	**795**	1150	1187	1413	**1416**	1644	1783	2141	**2168**
72	341	296	341	326	750	678	810	810	1203	1197	1430	**1454**	1768	1862	2136	**2215**
73	350	299	350	329	759	688	834	818	1229	1242	1462	**1549**	1768	1872	2197	**2252**
74	350	306	350	341	767	703	842	**845**	1229	1269	1512	**1562**	1774	1910	2291	**2299**
75	350	310	350	345	771	714	867	854	1274	1282	1530	**1571**	1862	1963	2303	**2365**
76	363	324	363	349	813	750	882	877	1300	1318	1569	**1597**	1935	2024	2387	**2415**
77	363	325	363	355	823	762	894	890	1308	1331	1591	**1631**	1947	2051	2433	**2456**
78	387	337	387	362	844	761	910	906	1382	1360	1621	**1654**	2014	2079	2479	**2502**
79	387	333	387	376	848	788	927	921	1382	1399	1639	**1668**	2037	2120	2498	**2550**
80	403	344	403	371	873	786	949	943	1395	1430	1691	1690	2037	2152	2554	**2597**
81	406	338	406	381	916	782	963	**972**	1406	1431	1730	**1731**	2064	2217	2648	**2665**
82	406	353	406	389	938	825	989	979	1475	1482	1742	**1773**	2141	2239	2680	**2715**
83	417	369	417	401	940	829	1002	**1006**	1501	1509	1762	**1804**	2141	2290	2696	**2752**
84	426	363	426	401	967	838	1021	1015	1534	1510	1818	**1825**	2229	2325	2790	**2803**
85	426	369	426	406	967	877	1043	1032	1552	1566	1866	**1871**	2232	2399	2819	**2877**
86	428	376	428	422	967	867	1053	1047	1573	1578	1882	**1890**	2375	2437	2875	**2929**
87	428	374	428	419	976	877	1073	1062	1598	1589	1934	1922	2375	2476	2913	**2988**
88	437	374	437	426	1050	890	1086	1079	1685	1629	1954	**1963**	2398	2513	2975	**3075**
89	443	378	443	432	1050	907	1102	1102	1690	1654	1990	**2009**	2400	2562	3067	**3104**
90	481	384	481	437	1060	940	1134	1116	1710	1696	2027	**2032**	2516	2633	3104	**3134**
91	481	393	481	450	1089	951	1134	**1146**	1748	1724	2031	**2054**	2516	2674	3143	**3192**
92	481	394	481	457	1089	966	1149	**1154**	1805	1750	2100	**2117**	2599	2729	3216	**3243**
93	481	402	481	456	1098	962	1171	1169	1813	1795	2130	2122	2604	2726	3283	**3305**
94	481	405	481	461	1124	986	1199	1198	1881	1811	2169	2165	2747	2788	3348	**3360**
95	481	413	482	474	1135	1010	1219	**1223**	1901	1846	2206	2202	2747	2817	3335	**3455**
96	509	414	509	475	1261	1023	1261	1228	1965	1863	2227	**2261**	2769	2911	3451	**3497**
97	515	419	515	489	1261	1027	1261	1245	1965	1899	2299	2269	2817	2960	3514	**3546**
98	531	429	531	486	1261	1055	1283	1266	1965	1929	2299	**2309**	2850	3001	3560	**3606**
99	531	449	531	497	1261	1040	1298	1290	2009	1950	2338	**2356**	2878	3043	3628	**3652**
100	554	451	554	502	1261	1074	1305	**1313**	2053	1975	2401	**2388**	3000	3117	3648	**3710**
Imp.				14				34				78				90
Time	145	500		164	61	181		507	267	152		767	108	520		1235

	$k=7$				$k=8$			$k=9$			$k=10$		
N	PD	SA	Web	ILS (ϕ)	SA	Web	ILS (ϕ)	SA	Web	ILS (ϕ)	SA	Web	ILS (ϕ)
2	7	7	7	7	8	8	8	9	9	9	10	10	10
3	7	13	13	13	14	14	14	18	18	18	19	19	19
4	16	21	21	21	26	26	26	28	28	28	33	33	33
5	16	32	32	32	40	40	40	43	43	43	50	50	50
6	29	47	47	47	54	54	54	61	61	61	68	68	68
7	31	61	61	61	70	70	71	80	80	81	89	89	90
8	46	79	79	79	91	90	91	101	101	102	114	114	114
9	47	92	92	93	112	112	113	126	126	128	141	142	143
10	68	110	110	111	130	131	133	154	154	157	172	172	174
11	69	128	129	132	152	152	154	178	178	181	206	206	209
12	95	150	152	155	176	177	181	204	204	209	235	235	240
13	95	174	178	181	202	205	210	232	235	242	267	268	275
14	119	204	219	217	228	236	243	265	268	278	298	305	313
15	129	211	220	223	257	273	280	296	309	318	337	347	358
16	155	238	241	249	286	317	326	330	352	358	378	393	406
17	161	256	266	272	312	332	332	367	396	405	415	442	458
18	186	281	291	298	344	361	368	398	451	466	458	496	509
19	195	305	323	326	370	390	398	438	469	472	498	554	569
20	226	332	349	360	403	425	434	472	506	517	542	625	641
21	236	361	380	393	438	463	471	517	548	559	592	650	650
22	270	384	418	425	467	501	508	555	595	614	643	691	704
23	273	410	448	454	501	542	549	596	640	651	685	747	750
24	308	444	481	492	538	585	595	639	690	699	739	800	818
25	350	467	520	531	583	626	637	688	739	752	792	857	875
26	365	499	548	570	612	664	688	726	791	810	854	910	931
27	382	526	585	599	648	712	738	780	840	859	896	976	1002
28	406	561	620	634	693	766	785	826	898	919	953	1041	1061
29	417	593	654	675	733	817	837	876	956	986	1015	1100	1132
30	458	620	691	714	787	849	897	925	1019	1041	1086	1173	1207
31	482	657	728	764	812	900	931	976	1104	1104	1138	1241	1275
32	518	695	778	803	866	966	976	1026	1139	1176	1194	1318	1351
33	537	723	814	844	900	1010	1037	1084	1201	1244	1253	1396	1436
34	561	751	851	891	945	1072	1089	1135	1270	1316	1329	1478	1514
35	586	811	914	934	1002	1113	1151	1190	1326	1398	1398	1555	1595
36	636	831	939	968	1042	1181	1205	1257	1405	1444	1459	1647	1679
37	668	863	976	1012	1079	1236	1272	1300	1477	1505	1516	1721	1761
38	709	923	1028	1055	1127	1286	1328	1367	1534	1577	1597	1790	1852
39	726	938	1084	1094	1192	1344	1397	1434	1609	1640	1665	1870	1987
40	786	970	1122	1148	1224	1416	1459	1489	1675	1728	1742	1946	2101
41	802	1016	1156	1197	1271	1496	1535	1562	1765	1793	1820	2058	2135
42	903	1064	1209	1249	1333	1526	1584	1639	1843	1871	1920	2149	2191
43	903	1112	1256	1301	1377	1597	1635	1683	1905	1957	1973	2224	2279
44	903	1140	1336	1340	1463	1653	1698	1752	1994	2042	2072	2319	2373
45	926	1192	1366	1408	1480	1723	1755	1820	2079	2126	2130	2415	2466
46	985	1243	1408	1448	1548	1794	1819	1906	2155	2220	2208	2507	2568
47	985	1268	1459	1521	1616	1847	1883	1958	2244	2312	2331	2600	2663
48	1054	1325	1531	1578	1658	1924	1957	2017	2336	2383	2387	2732	2760
49	1074	1356	1592	1649	1729	1989	2018	2103	2397	2470	2470	2828	2880
50	1113	1397	1639	1699	1772	2041	2089	2179	2492	2569	2556	2893	2991
51	1161	1450	1662	1744	1855	2132	2152	2243	2566	2637	2639	3006	3090
52	1231	1486	1734	1804	1888	2203	2218	2325	2686	2716	2745	3134	3202
53	1241	1537	1808	1886	1949	2234	2288	2429	2713	2798	2825	3261	3306
54	1288	1577	1856	1932	2006	2356	2383	2473	2805	2884	2892	3339	3412
55	1325	1639	1896	2000	2084	2429	2462	2570	2935	2996	3054	3452	3530
56	1358	1701	2003	2073	2162	2444	2533	2623	3021	3060	3100	3551	3643
57	1479	1721	2024	2098	2194	2554	2620	2704	3119	3162	3215	3651	3767
58	1479	1795	2043	2156	2258	2650	2679	2796	3187	3268	3305	3795	3843
59	1509	1821	2136	2187	2356	2733	2793	2881	3297	3350	3399	3889	3977
60	1577	1899	2232	2277	2393	2796	2873	2939	3420	3446	3500	4090	4109
61	1615	1928	2266	2316	2488	2868	2966	3021	3525	3565	3588	4158	4202
62	1680	2023	2345	2367	2541	2977	3048	3132	3636	3651	3700	4313	4322
63	1680	2035	2376	2417	2607	3056	3160	3215	3690	3760	3767	4355	4445
64	1769	2093	2452	2484	2734	3097	3207	3292	3820	3868	3955	4514	4560
65	1786	2132	2492	2547	2723	3219	3286	3357	3932	3991	4034	4581	4695
66	1857	2180	2543	2606	2841	3279	3418	3474	4004	4088	4143	4769	4818
67	1868	2238	2638	2672	2868	3399	3488	3543	4081	4200	4224	4942	4981
68	1940	2295	2693	2714	2956	3453	3600	3647	4212	4317	4360	4995	5077
69	1965	2351	2746	2794	3075	3520	3704	3716	4317	4400	4455	5127	5221
70	2130	2417	2838	2856	3130	3588	3779	3841	4464	4516	4539	5276	5366
71	2130	2451	2871	2939	3161	3749	3877	3936	4548	4666	4689	5437	5479
72	2177	2503	2960	2992	3220	3810	3962	4027	4666	4758	4812	5556	5625
73	2206	2598	3042	3077	3305	3932	4009	4134	4776	4858	4873	5661	5746
74	2244	2614	3120	3117	3432	3941	4127	4224	4915	4997	5038	5817	5879
75	2295	2703	3157	3230	3513	4073	4213	4298	5006	5141	5171	5937	6015
76	2375	2756	3218	3289	3559	4178	4326	4395	5179	5261	5254	6111	6163
77	2403	2819	3323	3359	3617	4266	4384	4492	5222	5364	5399	6272	6305
78	2505	2870	3387	3432	3684	4390	4491	4577	5385	5543	5489	6384	6449
79	2525	2950	3474	3488	3775	4465	4585	4705	5535	5631	5633	6466	6580
80	2590	2979	3550	3564	3877	4565	4695	4807	5577	5792	5773	6653	6733
81	2642	3086	3619	3638	4001	4679	4721	4888	5748	5922	5901	6780	6842
82	2753	3118	3669	3727	3998	4719	4809	5030	5859	6041	6013	6935	7041
83	2767	3195	3723	3800	4076	4848	4906	5102	5976	6196	6097	7094	7258
84	2838	3227	3870	3883	4183	4920	5006	5222	6119	6357	6273	7256	7362
85	2874	3299	3919	3954	4324	5032	5110	5340	6212	6479	6397	7357	7508
86	3103	3335	3958	4032	4397	5164	5205	5423	6346	6606	6491	7532	7687
87	3103	3450	4095	4119	4474	5225	5302	5538	6469	6761	6622	7639	7837
88	3183	3500	4166	4199	4524	5340	5426	5667	6660	6873	6803	7877	8022
89	3183	3541	4176	4290	4578	5450	5515	5774	6750	7004	6872	7950	8151
90	3190	3661	4308	4362	4699	5576	5608	5832	6901	7152	7040	8128	8325
91	3234	3677	4379	4423	4850	5626	5696	5969	6950	7296	7163	8330	8464
92	3277	3760	4428	4526	4873	5758	5822	6081	7067	7396	7286	8442	8681
93	3361	3811	4512	4574	4984	5832	5925	6231	7342	7446	7488	8601	8828
94	3474	3888	4581	4675	5067	6007	6032	6329	7436	7642	7536	8774	9066
95	3531	3940	4703	4758	5154	6064	6148	6396	7469	7748	7741	8877	9252
96	3639	4070	4808	4862	5220	6222	6227	6516	7645	7926	7777	9146	9445
97	3639	4069	4848	4919	5316	6304	6364	6649	7781	8011	8038	9379	9550
98	3690	4147	4936	5007	5445	6376	6467	6776	7896	8152	8242	9381	9820
99	3731	4214	4999	5117	5477	6448	6571	6912	8023	8258	8344	9617	9998
100	3903	4335	5040	5184	5597	6617	6692	6983	8228	8464	8450	9835	10233
Imp.				92			93			93			92
Time	232	246		698	460		846	470		1087	470		1166

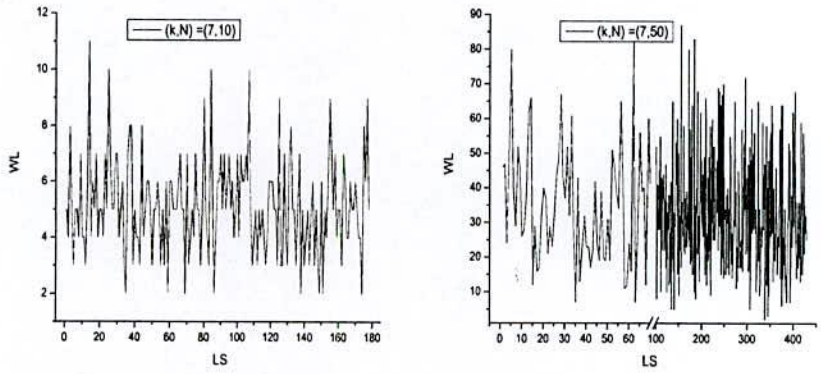

Fig. 5.7: The history of WL values for (a)$(k, N) = (7, 10)$; (b)$(k, N) = (7, 50)$ during LocalSearch

We will first discuss the time required by a local search. We do not discuss the time required by each swap move: this is the same as in ILS(D_1) and is at most $O(N)$. However, the number of swap moves which have to be attempted at each iteration is now different. Indeed, since we are considering the $\mathcal{LM}_{Rp\phi}$ local move, we have to consider all possible pairs of points (also those not involving critical points). Therefore, the number of swap operations is $O(kN^2)$. Note that this is an upper bound: since we are employing the FI acceptance criterion, we perform swap operations only until an improvement is observed.

Next, we need to derive some formula for the number of times the While-Loop is executed during a local search, i.e. for the number of iterations performed by a local search. As before, we will denote this number with WL. Note that with respect to ILS(D_1) we made a change in the local search, adopting the FI acceptance criterion rather than the BI one. Figure 5.7 shows the history

of WL values during different local searches for (a) $(k, N) = (7, 10)$) and (b) $(k, N) = (7, 50)$. We observe in Figure 5.7(a) that most of the time WL lies near 5 and never exceeds 12, whereas in (b) we notice that most of the time WL lies near 35 and the maximum value of WL is near 90. Therefore, it seems that WL increases together with N both for what concerns Average WL (AWL) values and Maximum WL (MWL) values. Figure 5.8 shows more clearly the relation between N and MWL as well as AWL. We observe that there is a linear impact of N. In order to establish the dependency of WL on k, we perform other experiments with $k = 2i : i = 1, 2, \cdots, 40$ and $N = 10, 25, 50, 75$. The relation between WL and k is not quite clear. Indeed, we observe in Figure 4.21 that WL is increasing with k for $k < 10$ but after that it decreases and finally tends to get stable around a constant value. It seems that by enlarging k the local search is able to reach a local minimum in quite few iterations with respect to lower values of k. This might be due to the fact that by increasing k we also enlarge the size of the neighborhood explored at each iteration of a local search. In what follows we will neglect the dependency of WL on k and

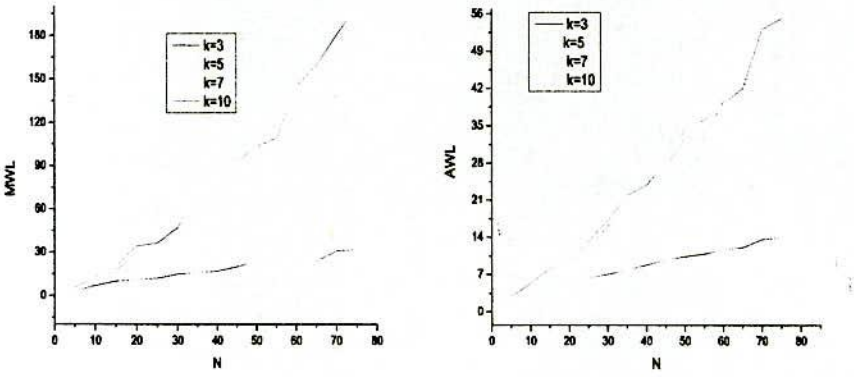

Fig. 5.8: The impact of N on (a) MWL (b) AWL

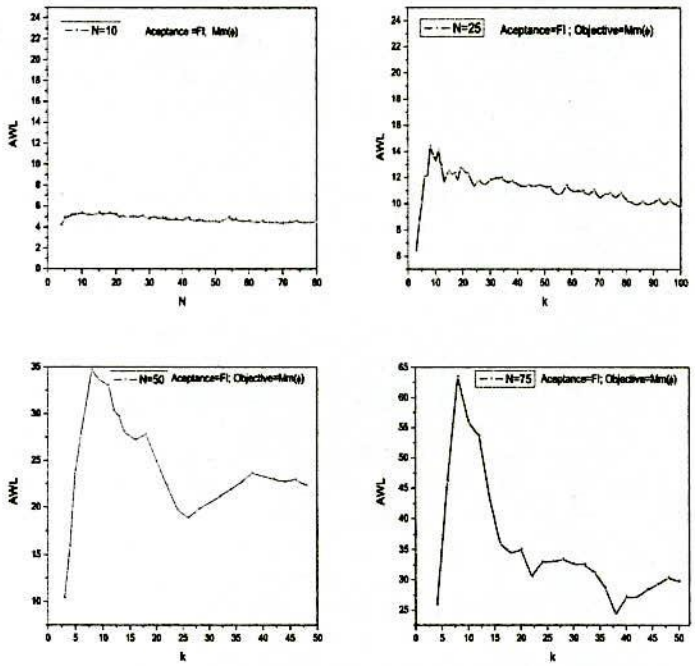

Fig. 5.9: The Impact of k on AWL

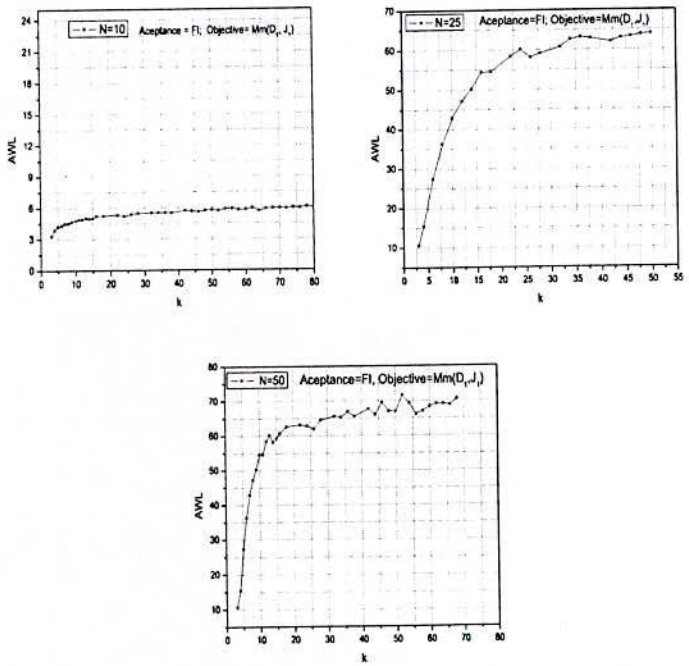

Fig. 5.10: The Impact of k on execution of AWL during LocalSearch with FI(First Improve) in Opt(D_1, J_1)

only consider WL as $O(N)$, but we should keep in mind that at small k values a dependency of WL on k is in fact present. Since when testing ILS(D_1) we employed the BI acceptance criterion, we would like to check, for completeness, if such phenomenon, i.e. the non dependency of WL on k at large k values, is somehow connected to the fact that we have considered the FI acceptance criterion. For this reason, we have performed another experiment with ILS(D_1) but with the FI acceptance criterion. We considered LHDs: $k = 2i : i = 1, 2, \cdots, 40$ with $N = 10, 50$. We observe in Figure 5.10 that WL increases quickly at small k values, while at large k values WL still increases, though more slowly. Such behavior is quite similar to the one observed in ILS(D_1) with the BI acceptance criterion.

If we put together the expected times for all the components of a local search, we can conclude that the approximate time required by a local search is

$$T \approx O(k^r N^q),$$

where we expect that the q value is close to 4, while the r value could range between 1 and 2. In order to find out the values of q and p experimentally, we performed the following experiments. At first we perform experiments to find the approximate value of q. For these experiments we considered $k = 5$ and $N = 20, 21, \cdots, 80$ and run ILS(ϕ) ten times for each LHD. In Figure

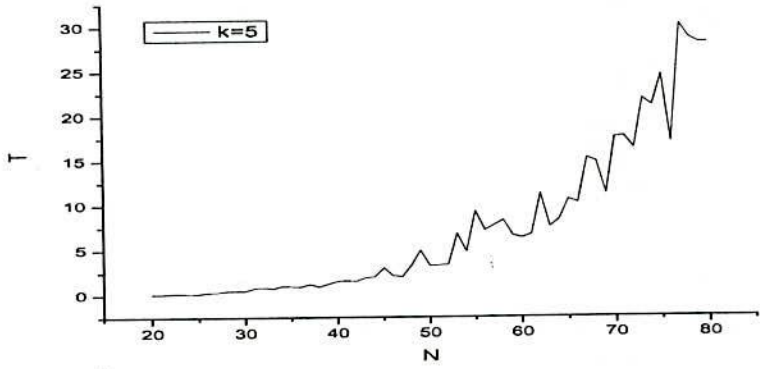

Fig. 5.11: Elapsed time per local search as a function of N

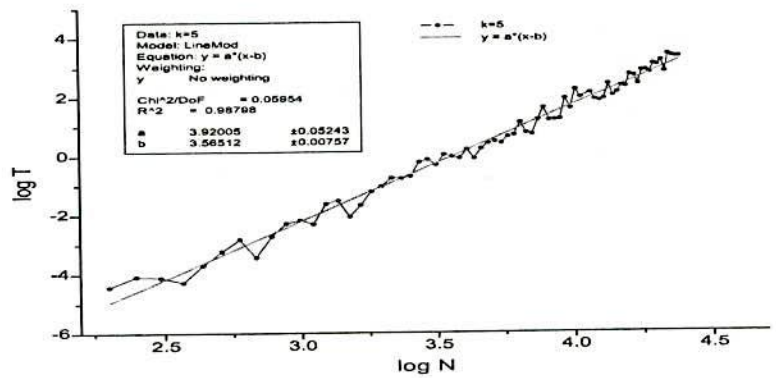

Fig. 5.12: Linear regression between $\log(T)$ and $\log(N)$

5.11, we plotted the average execution time per local search as a function of N. We observe that the increase with N is non linear. Therefore, we applied the logarithmic transformation

$$\log T = q(\log N - b),$$

where T denotes the average elapsed time, and then fitted the data in a linear regression. According to the data, we have that the value of q is 3.92 (see Figure 5.12), thus very close to the expected one, 4.

Now to find out the approximate value of r we performed experiments by considering LHDs with $k = 2i : i = 1, 2, \ldots, 25$ with $N = 50$. Figure 5.13 shows the impact of k on the average elapsed time per local search. In the figure we observe that T increases somewhat linearly with the increase of k. In order to find out the approximate time complexity with respect to k, we have fitted the data (see Figure 5.14) and detected a value of r approximately equal to 1.13,

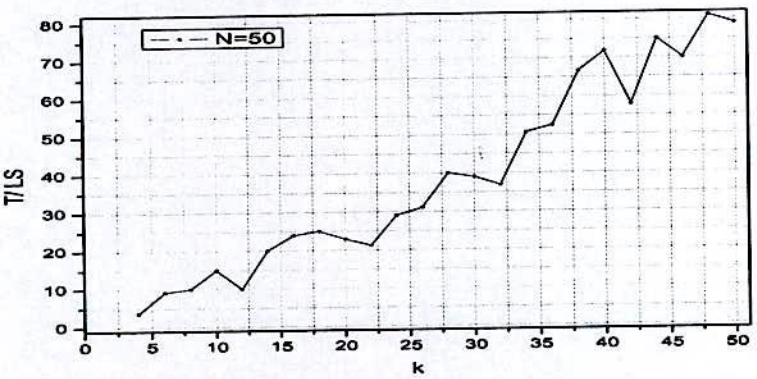

Fig. 5.13: Impact of k on T

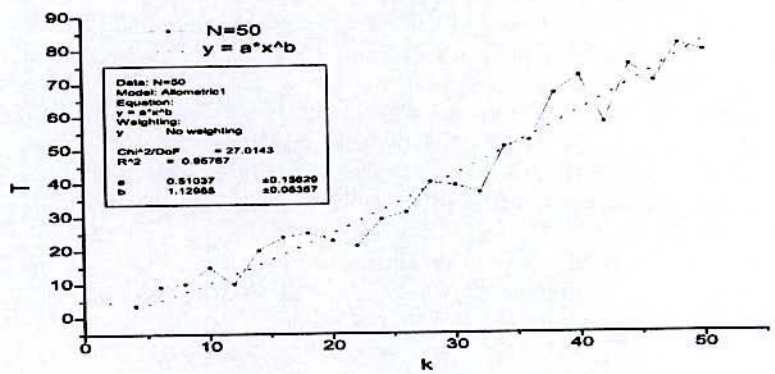

Fig. 5.14: The approximate time complexity of k for LS obtained by the experiments

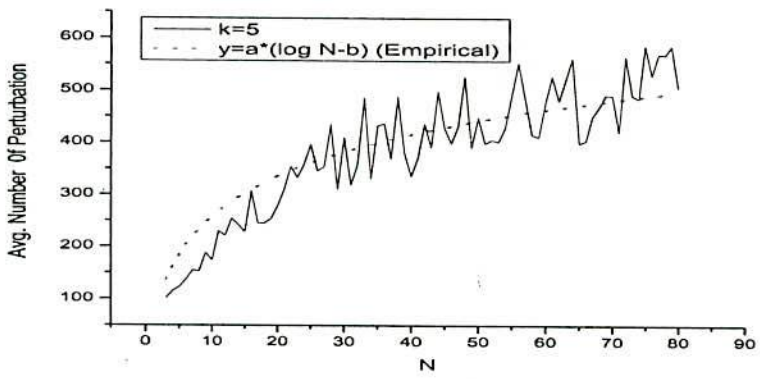

Fig. 5.15: Relation between the number of perturbations and N

again in accordance with what previously derived .

In order to derive the overall time complexity of ILS(ϕ) we still need to derive a formula for the number of perturbations (i.e. the number of local searches) performed during each ILS run. To find out the impact of N on such number

we considered LHDs with $k = 5$ and $N = 3, 4, \ldots, 80$ performing ten runs for each LHD. From the experiments (see Figure 5.15) we notice that there is a significant impact of N on the number of perturbations. We also try to establish a functional relation between N and the number of perturbations. Similarly to what already observed for ILS(D_1), the relation appears to be a logarithmic one with respect to N (see the dot curve in Figure 5.15). We point out that in both cases such logarithmic behavior is probably due to the fact that a fixed value for MaxNonImp (100 for ILS(ϕ), 1000 for ILS(D_1)) has been employed in all these tests, so that the total number of perturbations tends to get stable as we increase N.
To find out the impact of k on the number of perturbations, we considered LHDs with $N = 50$ and $k = 2i : i = 1, \ldots, 10$. From the experiments we notice that, in spite of a peak at $k = 6$, there is no significant impact of k on the number of perturbations invoked during a run (see the bar diagram in Figure 5.16).

In conclusion, summing up all the previous observations, we have that the time required for a single ILS(ϕ) run appears to be $O(k^r N^q \log(N))$, with r slightly larger than 1 and q slightly lower than 4.

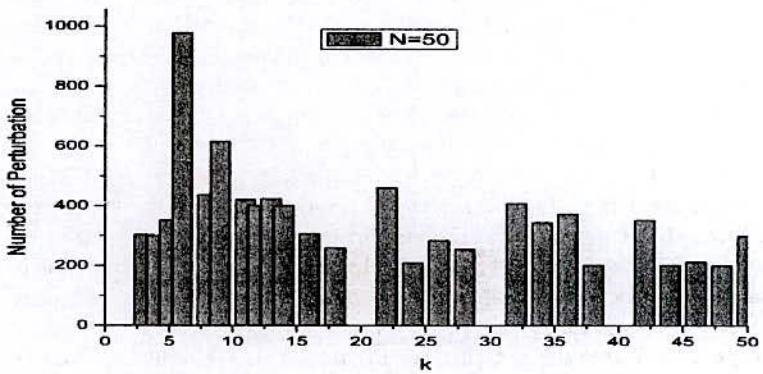

Fig. 5.16: Impact of k on the number of perturbations

6. PACKING PROBLEMS: DEFINITIONS AND MATHEMATICAL MODELS

The general problem of finding the densest packing of objects without overlapping in a bounded space is a classical one which has a wide spectrum of applications in scientific as well as engineering fields [56, 60, 61, 224, 225, 229, 230]. The packing problem consists of packing a set of geometric objects of fixed dimensions and shape into a region Ω of predefined shape, in such a way that the dimension of the region is as small as possible. In this thesis we consider two-dimensional packing problems. Moreover we focus on the special case where the n objects are identical (non-identical) circles, the region Ω is circular and the objective function is to minimize the radius of the region Ω. Therefore, we consider the Identical Circles Packing in a Circular Container (ICPCC in what follows) problem and the Non-Identical Circles Packing in a Circular Container (NICPCC in what follows) problem. If we denote by C the circular container, by r its radius, by C_i, $i \in I = \{1, 2, \ldots, n\}$ the n circles, and by r_i, $i \in I$, the radii of the n circles, NICPCC amounts at searching for the smallest radius r of C such that $C_i \subseteq C \; \forall \, i \in I$, and $C_i^0 \cap C_j^0 = \emptyset$ for all $i \neq j$, where C_i^0 denotes the interior of circle C_i (circles do not overlap). Of course, ICPCC can be viewed as a special case of NICPCC where $r_i = r_j$ for all i, j

6.1 Some definitions

If the positions of n circles are fixed, we call the set of positions a configuration. In the Cartesian coordinate system a configuration is denoted as

$$X = (x_1, y_1, \ldots, x_i, y_i, \ldots, x_n, y_n),$$

where (x_i, y_i) denotes the position of the center of circle i.

Definition 1: Given a configuration X, we say that two circles i, j *overlap*, if

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} < r_i + r_j$$

We also define the embedded depth Ed_{ij} between the i-th circle and the j-th circle as

$$Ed_{ij} = \max\{0, r_i + r_j - \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}\}$$

Similarly, we say that the i-th circle and the large container circle overlap (with radius r), if

$$\sqrt{x_i^2 + y_i^2} > r - r_i$$

The embedded depth Ed_{0i} between them is defined as

$$Ed_{0i} = \max\{0, r_i + \sqrt{x_i^2 + y_i^2} - r\}$$

Definition 2: Two circles i, j are in *touch (contact)* if the distance between their centers is equal to the sum of their radii, i.e.,

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} = r_i + r_j$$

Definition 3: A circle is said to be free if the center of this circle can be moved by a positive distance in some direction without causing overlapping with other circles and with the circular container. Note that if a packing contains one or more free circles then the solution is obviously not unique.

See also Figure 6.1 for a graphical illustration of the definitions.

6.2 Mathematical models

Although the NICPCC and ICPCC problems are geometrical ones, they can be easily reformulated as global optimization ones. A possible mathematical model for the NICPCC problem is the following:

$$\min \quad r \tag{6.1}$$

subject to

$$\sqrt{x_i^2 + y_i^2} \leq r - r_i \qquad i \in I \tag{6.2}$$

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \geq r_i + r_j \quad i, j \in I, \ \ i < j \tag{6.3}$$

$$LB_r \leq r \tag{6.4}$$

where $LB_r = \max_{i \in I} \ r_i$ (of course, ICPCC can be viewed as the special case of NICPCC where $r_i = 1$ for all $i \in I$). Constraints (6.2) indicate that each circle i is within the container. There are n such constraints, one for each circle (C_i) (that is $Ed_{0i} = 0$ for all $i \in I$). Constraints (6.3) guarantee the non-overlap condition for any pair of distinct circles (C_i, C_j) (that is $Ed_{ij} = 0$ for all $i, j \in I$, $i \neq j$). There are $n(n-1)/2$ such constraints. Constraint (6.4) provides a positive lower bound for the radius r of the container circle [106]. It substitutes the non-negativity constraint. The model makes the NICPCC problem unbounded if we eliminate this constraint. Then, the model has a total of $(1 + n(n+1)/2)$ constraints, and $2n + 1$ variables: Among the $2n + 1$ variables, $2n$ variables representing the coordinates $(x_i, y_i) : i \in I$ of the n circles, and one variable being the radius of the container circle C (whose center is assumed to be the origin).

The above model can be modified in such a way that we can get rid of the square roots. The equivalent model is the following

$$\min \quad r$$

subject to

$$\begin{aligned} x_i^2 + y_i^2 - r^2 + 2r_i r &\leq r_i^2 \qquad i \in I \\ (x_i - x_j)^2 + (y_i - y_j)^2 &\geq (r_i + r_j)^2 \quad i, j \in I, \ \ i < j \\ LB_r &\leq r \end{aligned} \tag{6.5}$$

This way the problem becomes a quadratic one (even with a linear objective function). Unfortunately, the quadratic constraints are "nasty" (non-convex) ones, thus making the problem a hard global optimization one with many local minimizers "hiding" the global one.

We finally remark that the optimal solutions of these problems need not be unique. For instance, if the optimal solution has free circles (see Definition 3), then we can move them around, thus obtaining an infinite set of solutions all with the same optimal radius of the container.

6.3 Problems equivalent to ICPCC

We conclude this chapter by observing that problem ICPCC is equivalent to a few other ones, namely:

- **Problem E-1**: Find the value of the maximum circle radius such that n identical non-overlapping circles can be placed in a unit circular container.
- **Problem E-2** : Locate n points in a unit circular container such that the minimum pair-wise distance d_n between any two points is maximal (maximin distance problem).
- **Problem E-3** : Instead of fixing the radius of the circular container and searching for the maximum radius of the circles in the packing, one can equivalently search for the minimum ratio of the radius of the container to the radius of the circles in the packing without fixing them.

For a given number n of circles, let r_n be the optimal value of problem **E-1**, d_n be the optimal value of problem **E-2**, and D_n the optimal value of problem ICPCC. Then, it is well known that the following relations hold between such optimal values (see, e.g., [88])

$$D_n = 1/r_n \qquad d_n = \frac{2r_n}{1 - r_n}, \qquad D_n = 1 + \frac{2}{d_n} \tag{6.6}$$

This is basically a consequence of the fact that given a collection of points in the unit circle at distance at least d from each other, the points can serve as the centers of a collection of circles of diameter d that will pack into a circle of diameter $1 + d$ as also illustrated in Figure 6.2.

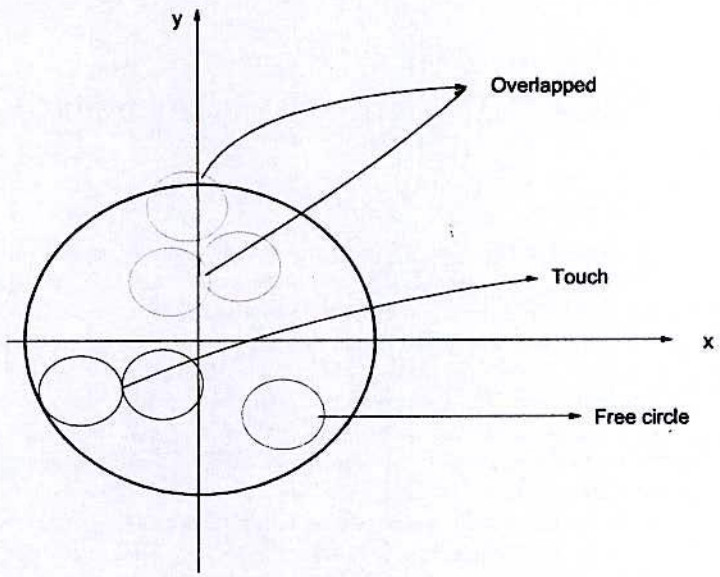

Fig. 6.1: Graphical illustration of the definitions

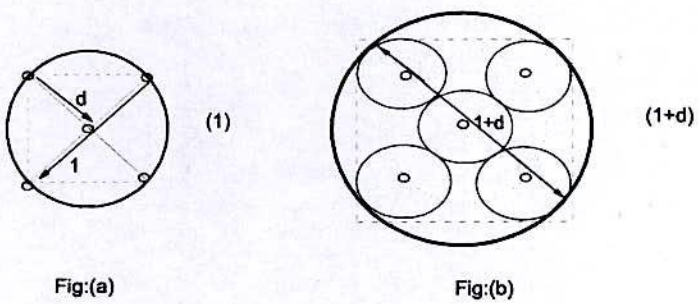

Fig. 6.2: Relation between points and circle packing

7. BASIN HOPPING ALGORITHMS FOR THE ICPCC PROBLEM

As a mathematical model for the ICPCC problem we will employ model (6.5) with $r_i = 1$ for all $i \in I$. For the sake of completeness, we point out that other models, like, e.g., (6.1)-(6.4), or any other model which can be obtained by any monotonic transformation of the objective function, are all *theoretically* equivalent to model (6.5) but might have a different *practical* impact on the performance of the algorithms (for a discussion about this subject we refer to [4, 52]). As already commented, the problem turns out to be a hard global optimization one, with the number of local minimizers tending to increase quite quickly with the number n of circles (this fact will be experimentally verified in Section 8.1). Such large number of local minimizers indicates that the simplest approach based on multiple local searches, Multistart , where we simply start different local searches from randomly generated initial points, is deemed to failure. As an alternative to Multistart here we are proposing a Monotonic Basin Hopping (MBH) approach.

7.1 MBH approach for ICPCC problem

The MBH approach is quite close to Multistart (they only differ in the mechanism for the generation of the initial points) but at the same time will also turn out to be dramatically more efficient than Multistart, at least for this problem. For ease of reference we report here the short pseudo-code of a MBH approach

Monotonic Basin Hopping

Step 1(Init): Let X_0 be randomly generated initial solution
Step 2: Let $X = \tau(X_0)$ be a local minimum
While **SR** not satisfied
  Step 3(PM): Let $Y := \zeta(X)$
  Step 4(LS): Let $X' := \tau(Y)$
  Step 5(AR): If $f(X') < f(X)$, then $X := X'$
  EndIf
EndWhile
Return X

The main ingredients of the method (highlighted in the code) are: an *Initialization* step (Init), a *Local Search* procedure (LS) denoted here by τ, a *Perturbation Move* (PM) denoted here by ζ, an *Acceptance Rule* (AR) and a *Stopping Rule* (SR).

In the next subsections we will detail our choices of Init, LS, PM, AR and

SR for the problem at hand.

7.1.1 Initialization

The initialization step is rather simple: we randomly generate an initial solution X_0 within a large enough region, and then we start a local search from it. Note that this is exactly what Multistart performs at each iteration. The difference in MBH is that only the first local minimizer is detected in this way, all the others are detected by local searches starting at points generated by the perturbation move.

7.1.2 Local search procedure

As shown in (6.5), our problem can be viewed as a non-convex one with objective and constraint functions continuously differentiable infinitely many times. Therefore, any local search method for this kind of problems can be employed. However, according to our experience, SNOPT [76] appears to be particularly well suited for these problems. Of course, constraint satisfaction in SNOPT (in particular for what concerns the non-convex non-overlapping constraints) can only be guaranteed within a given tolerance (we set such tolerance to 10^{-12} for all the experiments). However, we remark that even in case of slight infeasibility of a given solution, we can easily restore feasibility by multiplying each variable by an appropriate factor (slightly) larger than 1.

7.1.3 Acceptance rule

Although in the pseudo-code above, following the *monotonic* principle, we have only defined a rather simple acceptance rule (namely, accept a candidate configuration only if it improves the current one), we would like to point out here that, following other heuristic approaches, like simulated annealing, also non-improving moves (backtracking) could be accepted. In fact some sort of backtracking is advisable, since MBH tends sometimes to get trapped into local and not global minimizers, but, according to our experience, randomly restarting the search when no more progress is observed (see also the discussion about the stopping rule SR) seems already a quite reasonable option.

7.1.4 Perturbation move

The perturbation move is certainly the main ingredient of MBH. We have already discussed that a good move should guarantee that the structure of the current local minimizer is not completely disrupted by the perturbation. This way, the method does not simply perform a random search among the local minimizers (as in Multistart), but it moves between different but "close" local minimizers, performing a sort of meta-local search (a local search in the space of local minimizers). In the case of equal circles we propose three simple perturbation moves, based on uniform random perturbation of some or all the coordinates of each circle's center within some interval $[-\Delta, \Delta]$. The moves are called Full Jerk (FJ) , Random Partial Jerk (RPJ), and Fixed Partial Jerk (FPJ) and are briefly introduced below.

(a) Full Jerk perturbation move

The FJ perturbation move is rather simple – all the center of the circles are displaced by some random quantity uniformly sampled within an interval $(-\Delta, \Delta)$. The single parameter Δ, on which the perturbation depends, is of great importance. If Δ is too small, the starting point will be very likely in the basin of attraction of the current local minimizer (we are not disrupting at all the structure of the current local minimizer); on the other hand, if Δ is too large, the method becomes basically equivalent to a Multistart method (which disrupts the structure too much). In Section 8.3 we will further discuss the choice of Δ and perform experiments in order to select an appropriate value for it. The pseudo-code structure for the FJ move is as follows

Pseudo-code of FJ

Step 1: Let $Z = \{z_{11}, z_{12}, \ldots, z_{n1}, z_{n2}\}$ be a local minimum
do $i = 1$ to n
  do $k = 1$ to 2
    Step 2: select $\Delta z_{ik} \in (-\Delta, \Delta)$ randomly
    Step 3: set $z'_{ik} := z_{ik} + \Delta z_{ik}$
  End do
End do
return $Z' = \{z'_{11}, z'_{12}, \ldots, z'_{n1}, z'_{n2}\}$

(b) Random Partial Jerk perturbation move

The RPJ perturbation move is similar to FJ, the only difference being that not all the circle centers are perturbed but only a limited number of them, selected at random (the position of all the other circles is left unchanged). The pseudo-code of the RPJ technique is as follows

Pseudo-code of RPJ

Step 1: Let $Z = \{z_{11}, z_{12}, \ldots, z_{n1}, z_{n2}\}$ be a local minimum and set $Z' = Z$
Step 2: select $\Delta n \in (1, n)$ randomly;
Step 3: randomly select a set $\bar{I} \subseteq I$ of cardinality Δn;
do $i = 1$ to n
If $i \in \bar{I}$ then
  do $k = 1$ to 2
    Step 4: select $\Delta z_{ik} \in (-\Delta, \Delta)$ randomly
    Step 5: set $z'_{ik} := z_{ik} + \Delta z_{ik}$
  End do
End if
End do
return $Z' = \{z'_{11}, z'_{12}, \ldots, z'_{n1}, z'_{n2}\}$

(c) Fixed Partial Jerk perturbation move

The proposed FPJ perturbation move is a variant of RPJ where the number Δn of perturbed coordinates is not randomly selected but is fixed in advance. The pseudo-code structure of the FPJ technique is as follows

Pseudo-code of FPJ

Step 1: Let $Z = \{z_{11}, z_{12}, \ldots, z_{n1}, z_{n2}\}$ be a local minimum and set $Z' = Z$
Step 2: set $\Delta n \in (1, n)$ deterministically
Step 3: randomly select a set $\bar{I} \subseteq I$ of cardinality Δn;
do $i = 1$ to n
If $i \in \bar{I}$ then
    do $k = 1$ to 2
        Step 4: select $\Delta z_{ik} \in (-\Delta, \Delta)$ randomly
        Step 5: set $z'_{ik} := z_{ik} + \Delta z_{ik}$
    End do
End if
End do
return $Z' = \{z'_{11}, z'_{12}, \ldots, z'_{n1}, z'_{n2}\}$

It is worthwhile to remark at this point that the initial configuration produced by any PM operation may be (and, in fact, often is) unfeasible. But we can easily restore feasibility by multiplying each variable for a large enough factor (unless the quite unlikely case of two circle centers being the same point occurs), or, alternatively, we can simply start the local search LS from the unfeasible point, letting LS itself restore feasibility.

7.1.5 Stopping rule

Ideally we would like to stop a method as soon as no more progress can be expected. For the Multistart method, for which, under mild assumptions, it can be proved that it is able to detect the global minimizer with probability one if we allow for an infinite number of local searches, this would mean stopping when the global minimizer has been detected. Instead, a single run of MBH does not necessarily lead to a global minimizer and might get stuck into a local minimizer from which it is unable to escape. In such case what we can do is simply to restart MBH from a new random starting point, thus ending up with a sort of Multistart where local searches are substituted by MBH runs (as already commented in Section 7.1.3, the alternative is to introduce backtracking in the search by changing the acceptance rule AR in such a way that also non-monotonic moves are performed). In practice, if no special information is available, we are unable to stop though when we are really sure that no more progress will be possible. The best we can do is to stop when no improvement has been observed for a sufficiently large number of iterations (of course, this is just a heuristic rule with no guarantee that improvements are not possible any more). The number of iterations without improvements after which we stop MBH is denoted by the parameter `MaxNonImp`. The choice of this parameter is particularly important: we should not stop too early (which could mean that we are not patient enough to reach the global minimizer) or too late (which would mean a waste of computational effort). The choice of this parameter will be computationally investigated in Section 8.2.

7.2 Population Basin Hopping for ICPCC Problem

Each run of MBH follows a single path through the space of local minimizers. An alternative to MBH is Population Basin Hopping (PBH) [95], inspired by

the Conformational Space Annealing algorithm (see, e.g., [144]), in which the single path search is substituted by a multiple path search. During this search, members of the population collaborate with each other in order to guarantee *diversification* of the search and to avoid the *greediness* which might characterize a single path search. All components of MBH are present in PBH. The new ingredient in PBH is the *dissimilarity* measure $\mathcal{D}$. New parameters are N_p (the size of the population) and *dcut* (a threshold dissimilarity value). If we denote by $\mathcal{S}$ the space of the solutions at which we are interested (in ICPCC basically the local minimizers), the dissimilarity measure can be defined as the following function

$$\mathcal{D} : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}^+$$

which, for a given pair of solutions, quantifies the diversity between them. Ideally, given two solutions $X, Y \in \mathcal{S}$, $\mathcal{D}(X, Y)$ should be close to zero only if $X, Y \in \mathcal{S}$ are very "similar "and, in particular, equal to 0 only if they represent (modulo symmetries, rotations, translations, numbering of circles, and so on) the same solution. We allow the concept of similarity to be problem-specific; the only essential requirement we impose is that for similarity of a solution $X \in \mathcal{S}$ with itself, it must hold that $\mathcal{D}(X, X) = 0$ [95].

Given the dissimilarity measure, the pseudo-code for PBH is the following.

Population Basin Hopping

Step 0(Init): Let $\mathcal{X}_0$ be a set of N_p randomly generated solutions
Step 1(LS): Compute $\mathcal{X} = \tau(\mathcal{X}_0)$ (initial population)
While the stopping rule SR is not satisfied
Step 2(PM): Compute $X'_i := \zeta(X_i) : X_i \in \mathcal{X},\ i = 1, 2, \ldots, N_p$
Step 3(LS): let $\mathcal{Y} := \tau(\mathcal{X}') \ : \ X'_i \in \mathcal{X}',\ i = 1, 2, \ldots, N_p$ (pert. pop.)
Sequential Replacement: Repeat $Y_i \in \mathcal{Y},\ \forall\ i = 1, 2, \ldots, N_p$
Step 4 let $X_h \in \mathcal{X}$ such that $\mathcal{D}(Y_i, X_h)$ is minimum
Step 5 (AR): **if** $\mathcal{D}(Y_i, X_h) < dcut$ **and** $f(Y_i) < f(X_h)$ **then**
set $\mathcal{X} := \mathcal{X} / \{X_h\} \cup \{Y_i\}$
EndIf
else if $\mathcal{D}(Y_i, X_h) \geq dcut$ **then**
select $X_s \in \mathcal{X}$ such that $f(X_s)$ is maximum, and
if $f(Y_i) < f(X_s)$ **then**
set $\mathcal{X} := \mathcal{X} / \{X_s\} \cup \{Y_i\}$
EndIf
EndRepeat
EndWhile
Return $\mathcal{X}$

Basically, at each iteration: a set $\mathcal{Y}$ of new candidates is generated through the application of the perturbation move to each member of the population; each new candidate Y_k, $k = 1, \ldots, N_p$, competes either with the member X_h of the current population $\mathcal{X}$ most similar to it with respect to the dissimilarity measure $\mathcal{D}$ (if $\mathcal{D}(X_h, Y_k) \leq dcut$), or with the worst member X_s of the population (if $\mathcal{D}(X_h, Y_k) > dcut$, i.e., Y_k is dissimilar enough with respect to all

members of the current population); if it wins (i.e., if it has a better function value), it replaces X_h (or X_s) in the population for the next iteration. Note that MBH is, in fact, a special case of PBH where $N_p = 1$. There is a trade off between two conflicting objectives in choosing N_p. We have already outlined above the (possible) advantages of PBH: increasing N_p increases diversification and decreases greediness. On the other hand, increasing N_p also increases the computational effort per iteration. We will discuss appropriate choices for N_p in Section 8.5.

The local search procedure and perturbations techniques of the PBH approach are the same as those for the MBH approach. Each individual is independently perturbed and a local search starts at the perturbed point. The real difference in PBH is represented by the acceptance rule. A candidate replaces the member of the population with which it competes only if it has a better function value as in MBH, but the member with which it competes is not necessarily (and, in fact, often it is not) the member of the population whose perturbation led to the candidate. Formally, a candidate Y_i does not necessarily compete with its "father" X_i. This means that Y_i could enter the new population even if $f(Y_i) > f(X_i)$ (a *backtracking* move which is not allowed in MBH), but also that Y_i might not enter the new population even if $f(Y_i) < f(X_i)$ (this is called *hesitation* and might be profitable in order to avoid the drawbacks of a too greedy approach). The stopping rule SR is basically the same employed for MBH: we stop if the best member of the population does not change for a fixed number `MaxNonImp` of iterations. In the following subsection we discuss our choices for the dissimilarity measure and the *dcut* value.

7.2.1 Dissimilarity measure

Since the dissimilarity measure $\mathcal{D}$ is the core component of the proposed PBH approach, we will discuss below a couple of possible choices of such measures for packing problems. Note that in [95] there are several dissimilarity measures proposed for molecular conformation problems. For what concerns the choice of the *dcut* value, we adopted in our PBH algorithm a simple definition: it is equal to half the average dissimilarity within the initial randomly generated population.

(a) Distance dissimilarity measure

Let $X = \{(\alpha_{i1}, \alpha_{i2})\}_{i=1,\dots,n}$ and $Y = \{(\beta_{i1}, \beta_{i2})\}_{i=1,\dots,n}$ be two distinct local minimizers. Let $\rho_h(X)$ be the distance of circle h from the barycenter of the centers of all circles in the local minimizer X, i.e., if we move the barycenter to the origin

$$\rho_h(X) = \sqrt{\alpha_{h1}^2 + \alpha_{h2}^2}$$

and define $\rho_h(Y)$ in a similar way; let δ_X be the vector whose components are the distances $\rho_h(X)\ \forall\ h = 1, \dots, n$ ordered in a nondecreasing way, i.e., $\delta_X[1] \leq \delta_X[2] \leq \dots \leq \delta_X[k] \leq \dots \leq \delta_X[n]$ where $\delta_X[k]$ denotes the k-th component of the vector δ_X. Similarly for the local minimizer Y. Then, the

distance dissimilarity measure is defined as follows

$$\mathcal{D}(X,Y) = \sum_{k=1}^{n} |\delta_X[k] - \delta_Y[k]|. \tag{7.1}$$

(b) Objective-distance dissimilarity measure

The objective-distance dissimilarity measure is very similar to the distance measure dissimilarity but also takes into account the difference between objective function values. More precisely, we define the objective-distance dissimilarity measure as follows

$$\mathcal{D}(X,Y) = |f(X) - f(Y)| * \sum_{k=1}^{n} |\delta_X[k] - \delta_Y[k]|. \tag{7.2}$$

The reason for this slight modification is due to free circles. When a configuration X has free circles, then we can move them around thus obtaining different configurations with a positive distance dissimilarity but a null objective-distance one with respect to X.

8. COMPUTATIONAL EXPERIMENTS AND DISCUSSION ABOUT ICPCC PACKING PROBLEMS

In this chapter we discuss the computational experiments about ICPCC that we have performed both to analyze the properties of the problem under investigation and to select the components and the parameter values for MBH and PBH in an appropriate way. All the tests have been performed on a Pentium IV: 2.4 GHz Processor and 1GB Ram.

8.1 Number of local minimizers

Our first set of experiments aims at showing how the number of local minimizers increases with the number n of circles. In order to recognize *distinct* local minimizers we consider their objective function values (i.e., the radius of the container). We adopt a conservative criterion by declaring two local minimizers different if they have a large enough difference in their objective function values. Taking into account the precision of the local solver, the threshold value above which two local minimizers are considered as distinct ones on the basis of their objective function values has been fixed to 10^{-8}. Note that, according to this criterion, we may consider as equal also different minimizers. In spite of this, the increase in the number of distinct local minimizers turns out to be very quick.

Indeed, in Figure 8.1 we report the total number of distinct local minimizers which have been detected over 50,000 local searches starting from randomly generated (over a sufficiently large box) initial points for n up to 40. We, also, investigate the total number of distinct local minimizers for the same local search by considering as distinct local minimizers when their difference between the objective function values is above the threshold (a) 10^{-5} and (b) 10^{-11}. These are displayed in Figure 8.2. We remark that the overall trend of the number of local minimizers is quite similar, though the number clearly increases as the threshold decreases.

The trend of increase of the number of local minimizers is not a regular one, but at the same time quite clear, showing a rapid increase with the number of circles. This gives a clear indication that Multistart is most likely not an appropriate method to tackle this problem, which will be confirmed by the results reported in Section 8.4.

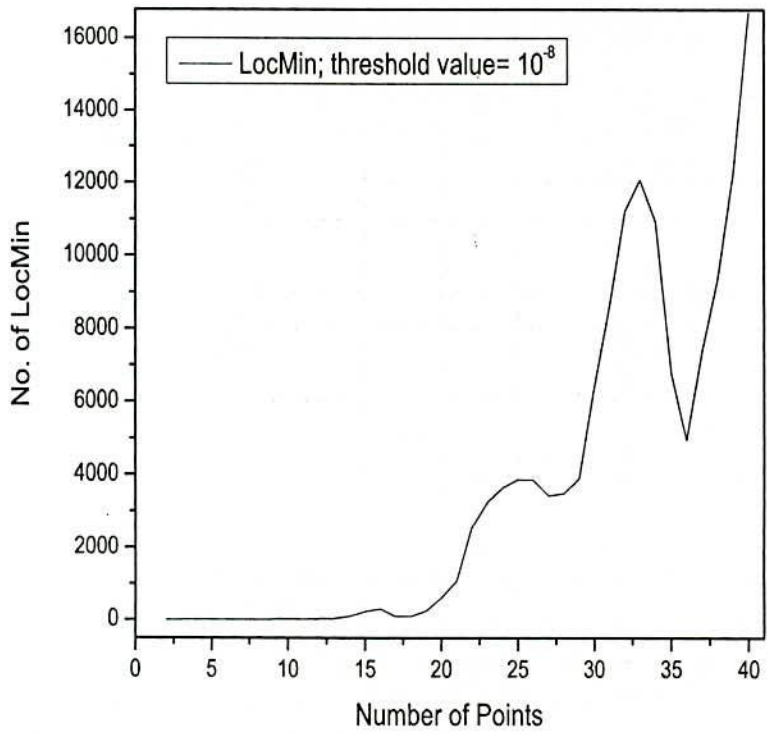

Fig. 8.1: Empirically determined number of local minima with the threshold value of objective function 10^{-8}.

8.2 *Choice of the stopping parameter (*`MaxNonImp`*)*

As already pointed out in Section 7.1, our stopping rule SR depends on the parameter `MaxNonImp` (sometimes also shortened to MNI in what follows). This parameter is an important one for MBH. Too low a value would cause to stop the algorithm before convergence is reached, while too large a value would cause a waste of computational effort. Also in this case we can hardly expect that there exists an optimal choice for all n values. So our aim is to look for a *robust* choice of this parameter value. For the experiments we consider the Full Jerk (FJ) perturbation technique with fixed $\Delta = 0.8$ (such choice for Δ will be justified by the following experiments). We tested the following set of values: MaxNonImp $\in \{50, 100, 200, 300, 400, 500\}$. The first set of results over 5 runs of MBH for each value $n = 30, 31, \ldots, 100$ is reported in Table 8.1.

The column `BestKnown` reports the best known solution according to [274] for a given n value. Column `OurResult` reports the best result we obtained over the 5 MBH(FJ) runs. In **boldface** we report the results which improve those

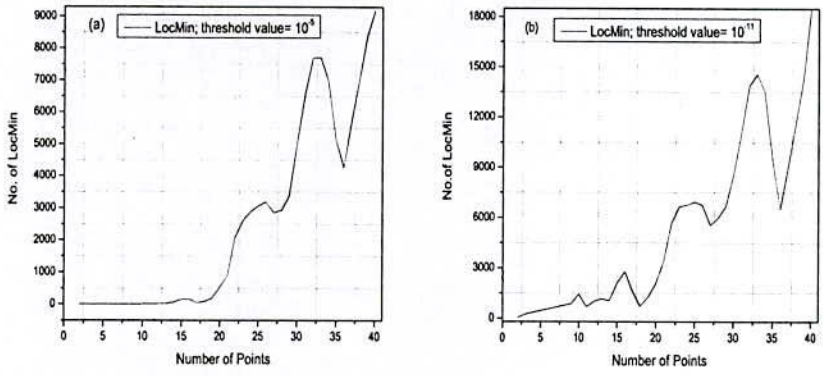

Fig. 8.2: Empirically determined number of local minima with threshold value of objective function (a) 10^{-5} and (b) 10^{-11}.

reported in [274], while in *emphasized* text we report the failures. In Column `NrSuccesses` we report for each `MaxNonImp` value the number of times the best solution reported in [274] has been reached (or improved). Finally, in Column `CPU time` we report the overall computation time (in seconds) for the 5 runs MBH(FJ), referred to the value `MaxNonImp`=500.

We remark that in 19 cases we could obtain improvements with respect to [274]. At the the same time, we also have some failures. In particular, we have 13 failures with `MaxNonImp`=50, but these immediately drop down to 9 with `MaxNonImp`=100 and progressively decrease to 5 with `MaxNonImp`=500. As a further test we decided to enlarge the number of MBH runs from 5 to 50 for the 9 cases where a failure occurred with `MaxNonImp`=100. The results are reported in Table 8.2. In what follows we denote these 9 cases as `Hard Instances` with respect to MBH(FJ). We notice that $n = 31$ turns out to be an extremely hard case for MBH(FJ): only with `MaxNonImp`=500 a single success could be obtained. This case will be further discussed in Section 7.2 where we will consider a different (and more successful over this instance) approach. In all the other cases we always have at least one success (with the only exception of the failure for $n = 83$ with `MaxNonImp`=50) and in three cases, namely $n = 78, 83$ and 92, we have further improvements with respect to [274].

In Figure 8.3 we have reported the total elapsed time (in hours) of 50 MBH runs with different `MaxNonImp` values, namely `MaxNonImp`$\in \{50, 100, 200, 300, 400, 500\}$ over the `Hard Instances`. We remark that though MBH(FJ) with `MaxNonImp` = 500 is always able to obtain the best known value, the computational effort is high compared to the MBH(FJ) approach with MaxNonImp=100 or 200. On the other hand, though the MBH(FJ) approach with MaxNonImp=50 is computationally cheaper than that with MaxNonImp=100, its performance is relatively poor.

Tab. 8.1: Overall results for 5 MBH(FJ) runs with different `MaxNoImp` values

			NrSuccesses						
n	BestKnown	OurResult	50	100	200	300	400	500	CPU time
30	6.197741070879	6.197741070879	5	5	5	5	5	5	110.49
31	6.291502622129	*6.352805480965*	0	0	0	0	0	0	130.14
32	6.429462970950	6.429462970950	5	5	5	5	5	5	151.14
33	6.486703123560	6.486703123560	5	5	5	5	5	5	164.69
34	6.610957090001	6.610957090001	5	5	5	5	5	5	174.41
35	6.697171091790	6.697171091790	5	5	5	5	5	5	179.81
36	6.746753793424	6.746753793424	5	5	5	5	5	5	247.81
37	6.758770483144	6.758770483144	5	5	5	5	5	5	240.7
38	6.961886965228	6.961886965228	5	5	5	5	5	5	213.43
39	7.057884162624	7.057884162624	5	5	5	5	5	5	231.1
40	7.123846435943	7.123846435943	5	5	5	5	5	5	283.29
41	7.260012328677	7.260012328677	4	4	4	5	5	5	264.59
42	7.346796406943	7.346796406943	4	5	5	5	5	5	294.11
43	7.419944856341	7.419944856341	5	5	5	5	5	5	449.45
44	7.498036682995	7.498036682995	4	4	5	5	5	5	313.27
45	7.572912326368	7.572912326368	0	2	3	3	3	3	482.36
46	7.650179914694	7.650179914694	5	5	5	5	5	5	414.0
47	7.724170052598	7.724170052598	3	5	5	5	5	5	428.22
48	7.791271430559	7.791271430559	4	4	5	5	5	5	467.42
49	7.886870958803	7.886870958803	1	1	1	1	2	2	775.54
50	7.947515274784	7.947515274784	2	2	4	4	4	4	655.69
51	8.027506952419	8.027506952419	5	5	5	5	5	5	542.55
52	8.084717190690	8.084717190690	5	5	5	5	5	6	699.28
53	8.179582826841	8.179582826841	1	2	3	3	3	3	807.84
54	8.203982383469	8.203982383469	2	3	3	3	3	3	701.1
55	8.211102550928	8.211102550928	3	3	4	4	4	4	1178.43
56	8.383529922579	8.383529922579	5	5	5	5	5	5	732.19
57	8.447184653410	8.447184653410	5	5	5	5	5	5	952.84
58	8.524553770140	8.524553770140	3	4	4	5	5	5	1078.31
59	8.592499959370	8.592499959370	5	5	5	5	5	5	1495.39
60	8.646219845458	8.646219845458	0	5	5	5	5	5	1168.12
61	8.661297575540	8.661297575540	5	5	5	5	5	5	1031.43
62	8.829765408972	8.829765408972	2	3	4	4	4	4	1400.12
63	8.892351537551	8.892351537551	3	4	5	5	5	5	1363.18
64	8.961971108486	8.961971108486	0	1	1	1	1	1	1266.97
65	9.017397323209	9.017397323209	3	3	3	3	3	3	1708.53
66	9.096665836768	**9.096279426924**	3	3	3	3	4	4	1813.23
67	9.169119588389	**9.168971881784**	1	1	2	2	2	2	2563.13
68	9.229773746751	*9.2340773401*	0	0	0	0	0	0	1390.52
69	9.269761266641	9.269761266641	5	5	5	5	5	5	1831.42
70	9.346055334486	**9.345653194048**	1	2	3	3	3	3	2391.79
71	9.416206538907	**9.415796896871**	4	5	5	5	5	5	2487.81
72	9.473890856713	9.473890856713	3	3	3	3	3	3	2246.99
73	9.540509504650	**9.540346152138**	1	1	1	1	1	2	2806.35
74	9.589239461626	**9.589232764339**	1	2	2	2	2	2	2543.33
75	9.672029634515	**9.672029631947**	1	2	2	2	2	2	3034.33
76	9.729596802162	9.729596802162	1	1	1	1	2	3	3927.92
77	9.798987497420	**9.798911924507**	1	1	3	4	4	4	3694.47
78	9.857712212603	**9.857709899885**	0	0	0	0	0	2	4852.32
79	9.905063467661	*9.909306621540*	0	0	0	0	0	0	3590.06
80	9.968151813153	*9.969802931195*	0	0	0	0	0	0	4032.13
81	10.010864241201	10.010864241201	2	2	2	3	3	3	5293.59
82	10.050824223451	10.050824223451	1	3	4	4	4	4	5432.51
83	10.116864426926	**10.116857875102**	0	0	1	2	2	2	7914.08
84	10.149530867236	10.149530867236	4	5	5	5	5	5	4780.79
85	10.163111465877	10.163111465877	3	4	4	4	5	5	7532.68
86	10.298701310984	**10.298701053110**	3	4	5	5	5	5	5128.9
87	10.363209161980	**10.363208505078**	2	5	5	5	5	5	4927.78
88	10.432342147160	**10.432337692732**	4	4	4	4	4	4	5578.01
89	10.500627671551	**10.500491814574**	2	2	2	3	3	3	4874.01
90	10.546069177954	10.546069177954	3	3	3	3	3	3	5059.66
91	10.566772233506	10.566772233506	2	3	3	3	3	3	6113.41
92	10.684689759023	*10.687984877108*	0	0	0	0	0	0	10041.7
93	10.733386127679	**10.733352600260**	0	1	2	3	3	3	7251.63
94	10.778032163883	**10.778032160252**	1	2	2	2	2	2	7831.68
95	10.840205021597	10.840205021597	0	0	0	0	1	1	13635.1
96	10.883669894312	10.883669894312	1	1	1	1	1	1	9701.68
97	10.938791648300	**10.938590110073**	1	1	1	2	2	2	9259.48
98	10.979383128207	10.979383128207	0	0	0	2	5	5	19099.9
99	11.037197388568	**11.035161062993**	4	5	5	5	5	5	7533.95
100	11.082527292540	**11.082149724310**	1	2	4	5	5	5	15311.6
Total improvement			16	17	18	18	18	**19**	
Total failure			13	9	8	7	6	**5**	

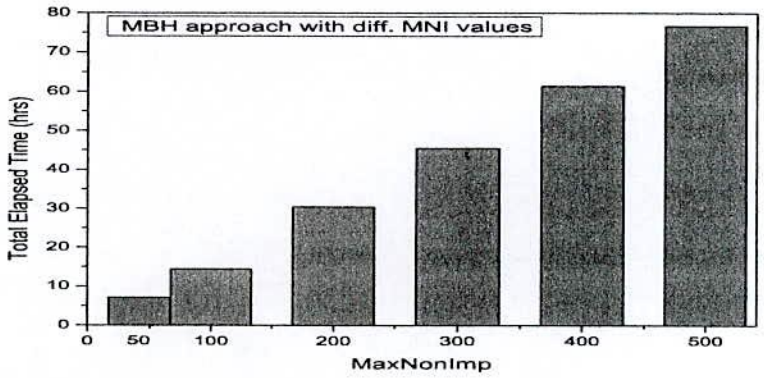

Fig. 8.3: The time comparison of 50 MBH runs with different MaxNonImp parameter values in some hard instances

Tab. 8.2: Overall results for 50 MBH(FJ) runs with different `MaxNoImp` values over some `hard instances`

			NrSuccesses					
n	BestKnown	OurResult	50	100	200	300	400	500
31	6.291502622129	6.291502622129	0	0	0	0	0	1
68	9.229773746751	9.229773746751	10	16	18	19	21	21
78	9.857712212603	**9.857709899885**	4	6	9	16	18	21
79	9.905063467661	9.905063467661	1	1	2	2	2	2
80	9.968151813153	9.968151813153	3	3	3	3	4	4
83	10.116864426926	**10.116857875102**	0	3	9	13	19	21
92	10.684689759023	**10.684645847916**	1	3	3	4	5	5
95	10.840205021597	10.840205021597	4	9	15	17	18	19
98	10.979383128207	10.979383128207	12	22	28	35	41	41

Tab. 8.3: Impact of `MaxNoImp` with respect to Number of Runs in MBH(FJ)

			NrSuccesses			
n	BestKnown	OurBestResult	R=50 MNI=50	R=25 MNI=100	R=13 MNI=200	R=5 MNI=500
60	8.64621984545799	8.646219845458	47	25	13	5
61	8.66129757554045	8.661297575540	50	25	13	5
62	8.82976540897208	8.829765408972	2	1	6	4
63	8.89235153755063	8.892351537551	28	15	11	5
64	8.96197110848576	8.961971108486	18	16	7	4
65	9.01739732320879	9.017397323209	40	2	0	1
66	9.09666583676771	**9.096279426924**	19	19	12	4
67	9.1691195883894	**9.168971881784**	3	2	5	2
68	9.22977374675067	9.229773746751	6	10	6	3
69	9.2697612666411	9.269761266641	3	3	2	3
70	9.34605533448604	**9.345653194084**	4	1	5	4
71	9.41620653890748	**9.415796896871**	4	6	10	5
72	9.47389085671311	9.473890856713	1	0	0	0
73	9.5405095046495	**9.540346152138**	1	2	3	3
74	9.58923946162617	**9.589232764339**	0	7	5	3
75	9.67202963451537	**9.672029631947**	2	4	4	2
76	9.72959680216164	9.729596802162	25	1	8	4
77	9.79898749742039	**9.798911924507**	0	0	3	3
78	9.85771221260262	**9.857709899885**	4	3	2	2
79	9.90506346766104	9.905063467661	0	0	0	0
80	9.96815181315344	9.968151813153	0	0	0	0
sum of successes			257	141	115	62
Total Num. of improvements			7	8	9	9
Total Num. of failures			4	4	4	3
Total elapsed Time (in hrs)			11.09	10.95	11.39	10.83
Total number of LS			56667	56375	59272	57022

One can also see things in a different way : what happens if we decrease the number of runs as we increase the value MaxNonImp in such a way that the overall effort is basically the same? In other words: are many short runs better than few long runs? For this reason we perform another experiment by considering MaxNonImp $= 50$, 100, 200, 500 with the corresponding number of runs $R = 50, 25, 13, 5$ respectively. For this experiment we only consider $n = 60, \ldots, 80$. We report the results in Table 8.3. We notice that, though the number of local searches and the total elapsed time are comparable, the performance of MBH(FJ) in terms of number of successes vs number of failures tends to get stable with MaxNonImp=100, which appears again as a reasonable choice.

As a general comment about the results we obtained, we notice that even a very aggressive strategy like choosing MaxNonImp=50 is often successful (of course, the number of successes is lower in this case, but this is counterbalanced by the lower computational effort). All the same with this choice failures occur more often (even over 50 runs). For this reason we believe that less aggressive choices like MaxNonImp=100 or MaxNonImp=200 represent the best compromise between the ability of reaching a good solution and the computational effort required.

8.3 The impact of the parameter Δ with different perturbation moves

We have already commented about the importance of parameter Δ. We remark that a good choice of Δ depends on two conflicting objectives. We define *successful* a run of MBH leading to a global minimizer. On one hand, we would like that a successful run of MBH converges to the global minimizer as fast as possible; on the other hand, we would like to maximize the probability that a run of MBH is successful. Note that the first goal is achieved when Δ is very small: in such case a successful run is just one where the initial point is already the global minimizer. On the other hand, this is exactly the situation where the second goal is lost. The second goal is achieved when Δ is very large (any run converges to the global minimizer in this case), but in this case the first objective is completely lost (the convergence is very slow, basically the same as the one of Multistart).

In this section we want to investigate computationally appropriate choices for the value Δ. The results of the experiments with the three proposed perturbation methods are reported in Tables 8.4-8.6. In such tables OurBestResult is the best overall result we obtained in our tests for a given n value (as usual, results in **boldface** denote improvements with respect to [274]).

As we have proposed three different perturbation strategies, we will investigate the impact of Δ for each strategy. For all these experiments, we consider $n = 60, 61, \ldots, 80$ and a number of runs $R = 5$. Our first experiment is with the perturbation strategy FJ. We tested four different values $\Delta \in \{0.6, 0.8, 1.0, 1.2\}$ in MBH(FJ). In all the tests we fixed MaxNonImp to 100 (as suggested by the previous experiments), but for $\Delta = 0.8, 1.2$ we also performed additional tests

Tab. 8.4: Impact of Δ in FJ perturbation based MBH Method

n	OurBestResult	Number of Successes in MBH (FJ)					
	(in MBH Method)	MaxNonImp = 100				MaxNonImp= 200	
		Δ =0.6	Δ =0.8	Δ =1.0	Δ =1.2	Δ =0.8	Δ =1.2
60	8.646219845458	3	5	5	3	5	5
61	8.661297575540	5	5	5	5	5	5
62	8.829765408972	0	1	2	0	3	1
63	8.892351537551	1	4	1	0	5	0
64	8.961971108486	1	3	0	2	3	3
65	9.017397323209	1	0	3	1	1	1
66	**9.096279426924**	1	4	0	0	5	0
67	**9.168971881784**	0	1	0	0	1	0
68	9.229773746751	3	1	0	0	1	0
69	9.269761266641	1	1	2	0	1	0
70	**9.345653194084**	0	2	0	0	2	0
71	**9.415796896871**	1	1	1	0	4	1
72	9.473890856713	1	0	3	0	1	0
73	9.540346152138	0	1	0	0	2	0
74	**9.589232764339**	0	1	1	0	2	1
75	**9.672029631947**	0	1	1	0	1	0
76	9.729596802162	0	1	4	0	3	0
77	**9.798911924507**	0	0	1	0	1	0
78	**9.857709899885**	0	0	0	0	0	0
79	9.905063467661	0	0	0	0	0	0
80	9.968151813153	0	0	0	0	0	0
No. of improvement		2	**6**	4	0	**7**	2
No. of success		10	**15**	12	4	18	7
No. of failure		11	**6**	9	17	3	14
Total elapsed time (in hrs)		3.02	4.06	6.73	8.55	4.62	10.44

Tab. 8.5: Impact of Δ in RPJ perturbation based MBH Method

n	OurBestResult	Number of Successes in MBH (RPJ)				
	(in MBH Method)	MaxNonImp(MNI) =100				MNI=200
		Δ =0.8	Δ =1.0	Δ =1.2	Δ =1.6	Δ =1.2
60	8.646219845458	5	5	5	4	5
61	8.661297575540	4	5	5	5	5
62	8.829765408972	0	1	0	1	3
63	8.892351537551	3	3	3	2	4
64	8.961971108486	2	2	1	1	1
65	9.017397323209	0	3	3	1	3
66	**9.096279426924**	2	1	4	0	5
67	**9.168971881784**	0	1	0	0	0
68	9.229773746751	0	0	1	1	2
69	9.269761266641	2	2	4	2	5
70	**9.345653194084**	2	3	2	1	2
71	**9.415796896871**	2	2	2	1	2
72	9.473890856713	**5**	3	1	3	5
73	9.540346152138	0	0	1	0	1
74	**9.589232764339**	**1**	1	**1**	**1**	2
75	**9.672029631947**	0	0	0	0	0
76	9.729596802162	0	0	2	1	4
77	**9.798911924507**	1	2	1	1	4
78	**9.857709899885**	0	0	2	0	2
79	9.905063467661	0	2	1	1	1
80	9.968151813153	0	0	1	0	2
No. of improvement		5	6	6	4	6
No. of success		12	15	18	15	19
No of failure		9	6	3	6	2
Total elapsed time(in hrs)		4.69	5.03	4.56	10.33	7.56

Tab. 8.6: Impact of Δ in FPJ perturbation based MBH Method

n	OurBestResult	Number of Successes in MBH (FPJ)							
	(MBH Method)	MaxNonImp (MNI)=100							MNI=200
		Δ =1.0	Δ =1.6	Δ =1.8	Δ =2.0	Δ = 2.2	Δ = 2.4	Δ =3.0	Δ = 2.2
60	8.646219845458	4	5	5	5	5	5	5	5
61	8.661297575540	5	5	5	5	5	5	5	5
62	8.829765408972	0	0	1	2	2	1	2	3
63	8.892351537551	5	2	2	3	3	3	3	5
64	8.961971108486	0	2	2	3	1	1	1	3
65	9.017397323209	0	1	3	5	5	1	2	5
66	**9.096279426924**	0	2	3	1	2	1	0	2
67	**9.168971881784**	0	1	0	0	0	0	0	0
68	9.229773746751	0	0	0	1	1	1	0	1
69	9.269761266641	1	0	2	1	1	1	0	1
70	**9.345653194084**	0	0	2	1	1	0	2	1
71	**9.415796896871**	0	2	2	0	1	1	0	1
72	9.473890856713	0	0	3	5	2	1	2	4
73	9.540346152138	0	0	0	0	0	0	1	0
74	**9.589232764339**	0	0	0	0	1	0	1	1
75	**9.672029631947**	0	0	0	1	1	1	1	3
76	9.729596802162	1	0	1	2	2	5	1	2
77	**9.798911924507**	0	1	2	2	1	0	0	1
78	**9.857709899885**	0	0	0	1	0	2	1	0
79	9.905063467661	0	0	0	0	0	0	0	1
80	9.968151813153	0	0	1	0	1	0	2	2
No. of improvement		0	4	4	5	6	4	4	6
No. of success		6	9	13	15	17	14	14	18
No of failure		15	12	7	6	4	7	7	3
T. elapsed time(hrs)		1.95	1.93	2.61	3.11	3.56	3.32	3.96	5.83

with MaxNonImp=200. All the results of this experiment are reported in Table 8.4. From the results reported in this table, it seems that for $\Delta = 0.8$, the MBH(FJ) approach performs relatively better compared to all other Δ values considered. Therefore, we may consider $\Delta = 0.8$ as a good value on average for MBH(FJ). Note that here we are searching for a robust choice for Δ, i.e. a choice working reasonably well for all n values. Indeed, the best possible choice for Δ usually varies with n.

For $\Delta = 0.8, 1.2$ we have also performed tests with MaxNonImp $= 200$. In particular, for $\Delta = 1.2$ this is justified by the fact that a larger neighborhood may require more iterations before detecting a better local minimizer. We notice that with MaxNonImp=200 the results do improve both for $\Delta = 0.8$ and for $\Delta = 1.2$ and that $\Delta = 0.8$ is still clearly the better option. Of course, as we increase MaxNonImp, we also increase the computational effort (see the last row of the table with the total elapsed time).

Our next experiment is with the perturbation strategy RPJ. We tested four different values $\Delta \in \{0.8, 1.0, 1.2, 1.6\}$ in MBH(RPJ). In all the tests we fixed the parameter MaxNonImp to 100, enlarging this value to 200 only for $\Delta = 1.2$ (the choice which will turn out to be the most robust one). All the results of this experiment are reported in the Table 8.5. From this table, it seems that for $\Delta = 1.2$, the MBH(RPJ) approach produces relatively better results compared to all the other Δ values considered. So $\Delta = 1.2$ appears as the most robust choice for the MBH(RPJ) approach (we point out again that we are looking for a robust choice, since the best possible choice for Δ usually varies with n) . When increasing MaxNonImp to 200 for $\Delta = 1.2$, the results are slightly improved compared to those with MaxNonImp $= 100$ (but recall that also the computational effort increases).

Finally, we performed another experiment for the perturbation strategy FPJ. We tested seven different values $\Delta \in \{1.0, 1.6, 1.8, 2.0, 2.2, 2.4, 3.0\}$ in MBH(FPJ). We fixed to 10% the percentage of circles, which are randomly perturbed in each iteration. In all the tests we fixed the parameter MaxNonImp to 100, but for $\Delta = 2.2$ (the choice which will turn out to be the most robust one), we also performed another experiment with MaxNonImp to 200. All the results of this experiment are reported in the Table 8.6. From this table, it seems that for $\Delta = 2.2$, the MBH(FPJ) approach produces relatively better results compared to all other Δ values considered and appears as the most robust choice. With MaxNonImp $= 200$, at the cost of a larger computational effort, the results with $\Delta = 2.2$ slightly improve.

We remark that, in all the experiments reported in Tables 8.4-8.6, the elapsed time tends to increase with the Δ value (for a fixed MaxNonImp value). This can be explained with the fact that as we increase Δ, we also increase the neighborhood to be explored, thus presumably increasing also the number of iterations needed before observing an improvement.

We also note that for Δ values larger than the most robust one for the given perturbation move, the performance tends to decrease as Δ increases. This is reasonable because, as we discussed earlier, a too large perturbation may de-

stroy the structure of the current local minimizer and the algorithm fails to preserve the "good" parts of the current configuration. Instead, for too small Δ values the performance tends to be poor because the algorithm can more easily get trapped in a configuration which is not a global minimizer.

Finally, we remark that the most robust choice for Δ is inversely proportional to the percentage of circles perturbed. Indeed, the FJ perturbation perturbs all the circles and its robust choice is 0.8, the RPJ perturbation perturbs on average 50% of the circles and its robust choice is 1.2, the FPJ perturbation perturbs 10% circles and its robust choice is 2.2. Basically, this means that when we perturb all the circles, in order not to destroy completely the structure of the current configuration we need to perform small perturbations of the circles, while as we decrease the number of the circles perturbed we may allow for larger perturbations still preserving the structure of the current one.

8.4 Comparison among different perturbation strategies

Tab. 8.7: Comparison among Different perturbations based MBH methods (with MNI=200, R=5) as well as MS approaches

n	BestKnown (Literature)	OurBestResult (MBH Method)	Number of Successes FJ (0.8)	RPJ (1.2)	FPJ (2.2)	MS (L)	MS (D)	BestResult (In MS)
60	8.646219845458	8.646219845458	5	5	5	4	7	8.64621
61	8.66129757554	8.661297575540	5	5	5	71	128	8.66129
62	8.829765408972	8.829765408972	3	3	3	1	2	8.82976
63	8.892351537551	8.892351537551	5	4	5	1	1	8.89235
64	8.961971108486	8.961971108486	2	1	3	0	0	8.96395
65	9.017397323209	9.017397323209	1	3	5	2	8	9.01739
66	9.096665836768	**9.096279426924**	5	5	2	0	0	9.09678
67	9.169119588389	**9.168971881784**	1	0	0	0	0	9.17621
68	9.229773746751	9.229773746751	1	2	1	0	0	9.23535
69	9.269761266641	9.269761266641	1	5	1	0	0	9.28787
70	9.346055334486	**9.345653194084**	2	2	1	0	0	9.34587
71	9.416206538907	**9.415796896871**	4	2	1	0	0	9.41689
72	9.473890856713	9.473890856713	1	5	4	0	0	9.47518
73	9.540346152138	9.540346152138	2	1	0	0	0	9.55517
74	9.589239461626	**9.589232764339**	2	2	1	0	0	9.61087
75	9.672029634515	**9.672029631947**	1	0	3	0	0	9.67643
76	9.729596802162	9.729596802162	3	4	2	1	1	9.72959
77	9.79898749742	**9.798911924507**	1	4	1	0	0	9.79926
78	9.857712212603	**9.857709899885**	0	2	0	0	0	9.85884
79	9.905063467661	9.905063467661	0	1	1	0	0	9.92100
80	9.968151813153	9.968151813153	0	2	0	0	0	9.97399
No. of improvement		8	7	6	6	0	0	
No. of success		21	18	19	17	6	6	
No. of failure		0	3	2	4	15	15	
Total elapsed time (hrs)			4.6	7.6	5.8	25.8	40.1	

The efficiency of the MBH approach strongly depends on the appropriate selection of the perturbation move. Here we will perform some experiments with the three different perturbation strategies that we have identified: (a) Full Jerk (FJ) (b) Random Partial Jerk (RPJ) and (c) Fixed Partial Jerk (FPJ). For the FPJ perturbation technique , we fixed to 10% the percentage of circles which are randomly selected to be perturbed. For these experiments we consider the number of circles $n = 60, 61, \ldots, 80$, we fix `MaxNonImp` to 200, we set $\Delta = 0.8, 1.2, 2.2$ for the MBH(FJ), MBH(RPJ) and MBH(FPJ) approaches

respectively (as suggested by the experiments in the previous section), and we fix to $R = 5$ the number of runs for each n value and perturbation technique. We report the results of the experiment in Table 8.7. In this experiment we also tested the Multistart (MS) algorithm. For a fair comparison, we allowed a number of local searches in MS (a) slightly larger than the largest number of local searches required by MBH with the three perturbation moves (we will denote these Multistart runs with MS(L) in what follows) (b) equal to twice the same number of local searches (we will denote these Multistart runs with MS(D) in what follows). The results obtained by the two MS approaches are also incorporated in Table 8.7.

We remark that with the different perturbation strategies, we obtained overall 8 improved results in the range $60 \leq n \leq 80$, namely $n = \{66, 67, 70, 71, 75, 77, 78\}$, with respect to those in [274] (as usual, in the table improvements are reported in **boldface**). Taking into account the overall results we have no failure (result worse than the one reported in [274]), although each single strategy has its own failures. We notice that: MBH (FJ) failed to obtain `OurBestKnown` result in three cases, namely $n = 78, 79, 80$, but with 7 (out of 8) improvements in the results; MBH(RPJ) has two failures, namely $n = 67, 75$, but with 6 (out of 8) improvements; MBH(FPJ) has four failures, namely $n = 67, 73, 75, 80$, and 6 improvements. Therefore, from the point of view of the quality of the results MBH(RPJ) appears as mildly superior with respect to the two other strategies, but the differences are not particularly significative. On the other hand, we notice from Table 8.7 that MBH(FJ) turns out to be computationally cheaper.

As largely expected, the results obtained with Multistart are usually quite poor, and are clearly inferior with respect to those obtained with MBH. In spite of the larger number of local searches allowed for MS, such algorithm is able to reach the best known solution only in few cases, and only in a single case (namely $n = 61$) the best known solution is reached quite regularly[1] (see also figure in Appendix B). We notice that MS(L), which performs a number of local searches basically equivalent to those performed by MBH with the three perturbation strategies is able to obtain only 6 successes, while MS(D) slightly improves MS(L) from the point of view of the number of times the best known solution is reached when a success occurs, but the overall number of successes as well as number of failures are same for both the cases.

Since the computational cost is an important factor when evaluating an algorithm, we have also incorporated the total elapsed time of each approaches in the Table 8.7. We notice that the total running time of MS(D) is about twice of that of MS(L), which is exactly what we expected. What was less expected is that the total elapsed time of MS(L) approach is about four times that of all the MBH approaches. We also display the time history of MS(L) and MBH(FJ) for the above experiments in Figure 8.4. Though the number of local searches in MS(L) and MBH(FJ) is almost equivalent, the running time of MS(L) oscillates between twice and eight times that of MBH(FJ) for each n. We also observe in that figure that, though not regularly, the gap tends to slightly increase with

[1] The case $n = 61$ belongs to the regular sequence $n = 3k(k-1)+1$, $k = 1, \ldots$, for which the (presumably) optimal solution has a circle centered at the origin and, around it, successive layers, each made up by $6j$ circles, $j = 0, 1, \ldots, k-1$

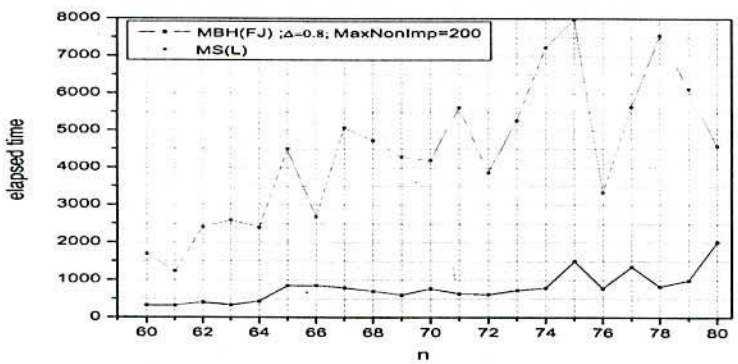

Fig. 8.4: Comparison of elapsed time (in second) between MS(L) and MBH(FJ) with respect to n

n. The reason for such gap is that MBH local searches are not started from completely random points as in Multistart, but in the neighborhood of a previously detected local minimizer, so that in the MBH approach, the local search procedure is able to converge in fewer iterations and the cost of a single local search is definitely lower.

n	BestKnown (In Literature)	OurBestResult (in MBH Method)	Number of Successes FJ (0.8)	RPJ (1.2)	FPJ (2.2)
67	9.169119588389	**9.168971881784**	12	2	0
73	9.54050950465	**9.540346152138**	11	2	2(s)
75	9.672029634515	**9.672029631947**	12	3	19
78	9.857712212603	**9.857709899885**	9	8	4
79	9.905063467661	9.905063467661	2	1	2
80	9.968151813153	9.968151813153	3	17	6

Tab. 8.8: Performance of MBH(FJ), MBH(RPJ) and MBH(FRJ) approaches (with MNI=200, $R = 50$)

In the above experiments reported in the Table 8.7 we observed that there are three failures ($n = 78, 79, 80$) in MBH(FJ), two failures ($n = 67, 75$) in MBH(RPJ), and four failures ($n = 67, 73, 75, 80$) in MBH(FPJ) over 5 runs with `MaxNonImp`= 200. Therefore, we have performed another experiment for $n = 67, 73, 75, 78, 79, 80$ with 50 runs and all the three strategies (`MaxNonImp`is still fixed to 200). The results are reported in Table 8.8. We observe that all the approaches are able to obtain `OurBestResults` except for the instances $n = 76$ and $n = 73$ in the MBH(FPJ) approach. In $n = 67$, 50 runs MBH(FPJ) are unable to obtain at least `BestKnown` value. Again in n=73, 50 runs MBH(FPJ) are unable to obtain `OurBestResults` but can obtain a better value than the

BestKnown value, namely $s = 9.540350146$. It is worthwhile to remark that with a slightly larger number of runs, MBH(FPJ) is able to obtain OurBestResults values which are better than the BestKnown value (see the table) for both the cases .

n	BestKnown (In Literature)	OurBestResult (in MBH Method)	Number of Successes FJ (0.8)	RPJ (1.2)	FPJ (2.2)
31	6.291502622129	6.291502622129	1	**24**	1
68	9.229773746751	9.229773746751	21	**37**	18
78	9.857712212603	**9.857709899885**	**21**	19	16
79	9.905063467661	9.905063467661	2	**7**	6
80	9.968151813153	9.968151813153	4	**20**	11
83	10.116864426926	**10.116857875102**	21	**23**	5
92	10.684689759023	**10.684645847916**	**5**	4	2
95	10.840205021597	10.840205021597	19	**20**	10
98	10.979383128207	10.979383128207	**41**	39	36
Total Success			135	193	105

Tab. 8.9: Comparison among MBH(FJ), MBH(RPJ) and MBH(FRJ) approaches in Hard Instances (with MNI=500, R=50)

Finally, we would like to compare the three strategies MBH(FJ), MBH(RPJ) and MBH(FPJ) over the previously defined nine Hard instances. We considered the number of runs $R = 50$ and we fixed MaxNonImp to the larger value 500. The results are reported Table 8.9. All strategies are able to reach the best known value at least once. Taking into account the overall number of successful runs, we observe that MBH(RPJ) performs best, while MBH(FJ) performs better than MBH(FPJ). Therefore, in spite of the fact that the MBH(RPJ) approach previously appeared as the most time consuming (see Table 8.7), it also appears as a quite robust one, guaranteeing a large number of successes also on the hardest instances.

8.5 Experiments with the PBH approach

8.5.1 Comparison with MBH on hard instances

In the first experiment we compare the behavior of PBH and MBH on the previously identified Hard Instances for MBH. We might think that the difficulty of such instances is due to the existence of different funnels, so that many runs of MBH are needed before hitting the (putative) global optimum. In this case the multi-path search performed by PBH should allow to detect the solution more easily, though at a higher computational cost (approximately, a single run of PBH has a cost which is N_p times larger than a single run of MBH, where N_p denotes the size of the population). We will compare MBH(FJ) and PBH(FJ) setting $\Delta = 0.8$ and MaxNonImp = 500 in both cases, setting $N_p = 10$ and employing the distance dissimilarity measure in PBH. In order to have a comparable overall computation time, we perform 50 runs of MBH and 5 of

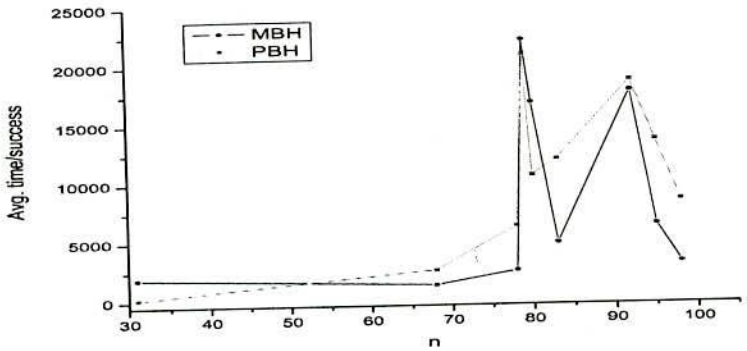

Fig. 8.5: Comparison between MBH and PBH regarding average elapsed time per success in some hard instances

PBH. The results are displayed in Table 8.10, where for each instance we report the percentage of successes. The results reported in the table suggest that PBH with a relatively large N_p value is certainly a robust approach, able to detect with a high percentage of success (often 100%) the solution of the hard instances. On the other hand, we should recall the higher computational cost of a PBH run.

Tab. 8.10: Comparison between MBH and PBH with $N_p = 10$ approaches in some hard instances

n	OurBestResult (in PBH)	Success (in %) PBH($N_p = 10$)	MBH
31	6.291502622129	**100**	2
68	9.229773746751	**100**	42
78	**9.857709899885**	**100**	42
79	9.905063467661	**60**	4
80	9.968151813153	**80**	8
83	**10.116857875102**	**100**	42
92	**10.684645847916**	**60**	10
95	10.840205021597	**80**	38
98	10.979383128207	**100**	82

For this reason, we compare the two approaches on the basis of the *elapsed time per success*. Figure 8.5 displays the average elapsed time per success of MBH(FJ) and PBH(FJ) on the hard instances. The figure shows that, with the remarkable exception of the $n = 31$ case, where PBH strongly outperforms MBH, the two approaches are often comparable but MBH is, usually, slightly superior.

8.5.2 Impact of population size N_p in PBH

In the previous experiments we considered PBH(FJ) with $N_p = 10$. Now we would like to investigate more thoroughly the impact of the population

size in PBH. In these experiments we consider PBH(FJ) with population sizes $N_p = \{1, 2, 4, 8, 10\}$. We set `MaxNonImp` $= 100$, $\Delta = 0.8$, and employ the `distance` dissimilarity measure. The experiments are performed on the large instances $n = 80 \ldots 100$. Note that $N_p = 1$ corresponds to the MBH approach. In order to have a comparable computation time, the number of runs is $R = 50, 25, 13, 6, 5$ for $N_p = \{1, 2, 4, 8, 10\}$ respectively. The results are reported in Table 8.11 in form of percentage of successes.

Tab. 8.11: The Impact of Number of Populations in PBH approach

	OurBestResult	Success (in %) with MNI= 100				
n	(in PBH)	N_p=1	N_p=2	N_p=4	N_p=8	N_p=10
80	9.96815181315344	4	8	25	50	100
81	10.0108642412007	38	68	83	100	100
82	10.0508242234505	58	92	100	100	100
83	**10.116857875102**	4	4	25	67	60
84	10.1495308672362	100	100	100	100	100
85	10.1631114658768	100	100	100	100	100
86	**10.29870105311**	72	100	100	100	100
87	**10.363208505078**	18	100	100	100	100
88	**10.432337692732**	74	100	100	100	100
89	**10.500491814574**	28	68	75	50	100
90	10.5460691779537	68	100	100	100	100
91	10.5667722335056	64	100	100	100	100
92	**10.684645847916**	0	0	0	17	0
93	**10.73335260026**	18	12	25	17	20
94	**10.778032160252**	36	28	42	50	60
95	10.840205021597	0	40	50	100	60
96	**10.8832027597222**	0	4	0	0	0
97	**10.938590110073**	14	4	42	67	100
98	10.979383128207	4	100	100	100	100
99	**11.0331411514456**	0	16	50	83	100
100	**11.08214972431**	18	64	83	100	100
Total No. Failure		4	1	2	1	2
Number of 100 % success		2	8	9	12	15

The results somehow confirms those in the previous subsection: indeed, in spite of one or two failures, the largest tested N_p values, say $N_p \in \{8, 10\}$, usually guarantee the highest percentage of successes (very often 100% successes), confirming that for large N_p values PBH turns out to be a quite robust approach. On the other hand, in many cases also small N_p values (even $N_p = 1$, i.e. MBH, although this is also the case with the largest number, 4, of failures) quite often guarantee a high percentage of successes (at a lower computational cost per success with respect to large N_p values). Basically, it seems that for these problems single or few path searches are often already quite efficient and that the benefits coming from the greater diversification guaranteed by PBH with larger N_p values are overridden by the larger computational cost per iteration. It is worthwhile to remark that we could obtain two further improvements at $n = 96, 99$ (see Figure 8.11)

8.5.3 Comparison of different dissimilarity measures

Since we have previously proposed two dissimilarity measures, we would like to perform a final experiment to compare the performance of PBH(FJ) with the two dissimilarity measures Distance Dissimilarity (DD) and Objective-Distance Dissimilarity (ODD). For this experiments we consider the instances $n = 80 \dots 100$ plus the hard instances with $n < 80$, set `MaxNonImp` $= 200$ and 500, $\Delta = 0.8$. We also consider three population sizes $N_p = \{2, 5, 10\}$ and always perform $R = 5$ runs. The results are displayed in Table 8.12. We notice that the differences between the two dissimilarity measures are not particularly significative, although, with the only exception of $N_p = 10$ and `MaxNonImp`=200, DD usually has a slightly lower number of failures and higher number of improvements. As a final remark, we point out that DD and ODD are reasonable measures but certainly not the only possible ones. A possible aim for future researches is that of proposing and testing new measures.

Tab. 8.12: The Comparison between different dissimilarity measures in PBH approach with $N_p = 2, 5, 10$. Note that in this table OurBestResult is denoted as OBR

	OBR	No. of Success for R=5 & MNI=200						No. of Success for R=5 & MNI=500					
		$N_p = 2$		$N_p = 5$		$N_p = 10$		$N_p = 2$		$N_p = 5$		$N_p = 10$	
n	(in PBH)	DD	ODD	DD	ODD	DD	ODD	DD	ODD	DD	ODD	DD	ODD
31	6.291502	1	0	3	0	5	1	2	0	5	1	5	3
68	9.229773	1	1	4	3	4	5	1	2	4	4	5	5
78	**9.857709**	1	1	3	5	2	4	3	3	5	3	3	5
79	9.905063	0	0	2	0	2	0	1	1	3	0	3	1
80	9.9681518	0	0	1	1	2	1	1	1	2	2	3	3
81	10.010864	4	4	3	4	5	4	4	4	4	5	5	5
82	10.050824	2	3	3	2	4	5	4	5	5	5	5	5
83	**10.116857**	0	0	2	2	0	1	1	1	4	4	3	3
84	10.1495308	4	5	5	5	5	5	5	5	5	5	5	5
85	10.163111	4	3	5	5	5	5	5	4	5	5	5	5
86	**10.29870**	3	4	4	2	4	5	5	5	5	5	5	5
87	**10.363208**	5	5	4	4	5	5	5	5	5	5	5	5
88	**10.432337**	4	5	5	5	5	5	4	5	5	5	5	5
89	**10.500491**	2	3	1	2	3	5	3	4	5	5	4	5
90	10.5460691	5	4	5	5	5	5	5	4	5	5	5	5
91	10.5667722	4	4	5	5	5	5	5	4	5	5	5	5
92	**10.684645**	1	0	0	1	0	0	1	1	1	1	0	0
93	**10.733352**	1	0	1	0	1	2	1	1	1	2	2	4
94	**10.778032**	0	0	1	0	0	0	0	0	2	0	0	0
95	10.840205	1	0	1	3	2	3	2	1	2	4	5	4
96	**10.883202**	0	0	0	1 (s)	0	1	0	0	1	1(s)	1	1
97	**10.938590**	0	1	2	1	1	2	1	3	3	5	4	5
98	10.979383	2	2	4	3	4	4	4	4	5	5	5	5
99	**11.033141**	1 (s)	1(s)	1	3(s)	2	3	1	1	1	4(s)	4	3
100	**11.082149**	2	2	2	3	2	4	3	4	4	5	5	5
T. Failure(0)		6	9	2	3	4	3	2	3	0	2	2	2
T. Impro. (12)		8	7	11	9	9	11	10	10	13	10	11	11
T. time (hrs)		48	44	62	76	117	107	120	111	152	179	297	269

9. PACKING PROBLEMS: NON-IDENTICAL CIRCLES IN A SMALLEST CIRCULAR CONTAINER

In Chapter 6 we have given the mathematical formulation for the problem of packing equal/unequal circles into a circular container and also proposed algorithms for solving the problem with equal circles, called Identical Circles Packing in a Circular Container (ICPCC) problem. In order to deal with the case of unequal circles one may think to extend the approaches employed for the case of equal circles with a slight variant in the perturbation moves: for instance, for the FJ perturbation strategy the coordinates of each circle i are displaced by a uniform random perturbation within the interval $[-\Delta r_i, \Delta r_i]$, where r_i denotes the radius of circle i (for RPJ and FPJ the displacement is restricted to a subset of circles). But, as we will see through some experiments, this simple extension is not the best way to tackle the problem. Indeed, the case of unequal circles has some peculiarities which have to be taken into account. The combinatorial side of this problem, represented by the different radii of the circles, can (and actually should) be exploited in some ways. In particular, we will propose a further possible perturbation move which is only suitable for the unequal circle packing problem. Moreover, we will also propose another strategy, again only suitable for unequal circles, where we first optimize a fraction of relatively larger circles, and then insert one or a part of the remaining smaller circles sequentially and simultaneously optimize them. All these issues, together with some computational experiments will be discussed in the following sections.

9.1 Proposed Sequential Insertion Based MBH

At first we discuss the strategy based on first removing and later re-inserting "small" circles. The basic idea is that, once a configuration with large circles is available, we can easily find some room for the smaller circles within the circular container without having to enlarge the radius of the container, or by only mildly enlarging it. Having removed "small" circles, we have the advantage of dealing with a smaller and simpler problem.

The technique is rather simple. First we select some circles to be removed; then, we apply MBH (or PBH) on the reduced set of circles; finally, the algorithm sequentially inserts the missing circles (following a non increasing order of the radii). In what follows we define this approach as Sequential Insertion Based MBH or PBH (SIB-MBH or SIB-PBH). The new procedure, which exploits the different radii of the circles, performs the following steps:

- (a) apply the Removal Strategy to remove "small " circles;
- (b) apply MBH (or PBH) on the remaining subset of larger circles;

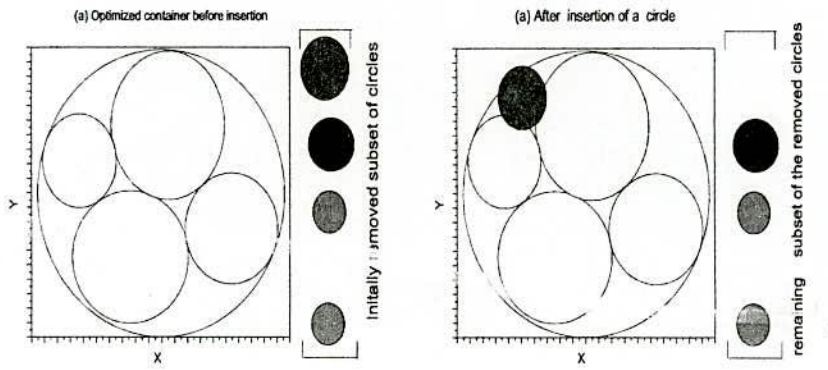

Fig. 9.1: Illustration of Insertion Rule

- (c) apply Insertion Rule for sequentially inserting the missing circles.

Besides all components of the MBH (or PBH) approach, there are two further components in this new approach namely (i) Removal Strategy (RemS) and (ii) Insertion Rule ($\mathcal{I}_{\mathcal{R}}$). Before giving a formal description of SIB-MBH, we describe these two new components.

(i) Removal Strategy (RemS). It is observed from the experiments that small circles are sometimes relatively easily inserted in holes of optimized configurations for some subset of the larger circles without having to enlarge the container or with just a small enlargement. According to our Removal Strategy, circles are indexed in decreasing order with respect to their radius. Then, a fraction of circles – "small" circles – is removed.

Of course, we need to define what a "small" circle is. We define a circle i as a small circle if its radius is at least four times smaller than the largest one, i.e., circle i is small if

$$r_i \leq \frac{1}{4} \max_{j=1,\ldots,n} r_j. \tag{9.1}$$

Let us denote the set of initially removed circles as $\mathcal{S}_{\mathcal{R}}$; then

$$\mathcal{S}_{\mathcal{R}} = \left\{ i \ : \ r_i \leq \frac{1}{4} \max_{j=1,\ldots,n} r_j \right\} \tag{9.2}$$

This strategy strongly simplifies some instances of the problem through a considerable reduction of the search space during the first phase where some circles are removed.

(ii) Insertion Rule ($\mathcal{I}_{\mathcal{R}}$). In the insertion process of a given circle c_s in $\mathcal{S}_{\mathcal{R}}$, first the algorithm creates a regular grid of points over a square region containing the circular container. The step of the square grid is half of the inserted

circle's radius. The edge length of the square region is the sum of the diameter of the container and the radius of the circle to be inserted, so that the circular container, which is optimized previously by the reduced circles, is fully enclosed within the square (both have the origin as their common center). Next, the algorithm searches for "free" spaces where to insert circle c_s. Given a point (x_i, y_i) over the grid, we declare the space around it as free, if its distance from the other circles' centers is at least equal to r_s, the radius of the circle to be inserted. In other words, if we place circle c_s with center in point (x_i, y_i) over the grid, the other circles' centers are not in the interior of such circle. Note that at least one free space certainly exists. Indeed, according to the above definitions, all the corners of the square certainly correspond to free spaces. It is worthwhile to note that the definition of "free"space does not mean that the space is large enough to contain circle c_s with no overlap with the other circles: a partial overlap is permitted and, actually, if the circle to be inserted is small compared to other circles, then even full overlapping may occur during the insertion process. It may also happen that the new circle is not fully (or even not at all) contained in the circular container. In spite of this partial or full overlap with other circles and of the possibility of crossing the border of the circular container, a local search procedure started at the new configuration with the added circle is able to adjust it in such a way that no overlap occurs without enlarging the radius of the circular container or with an as small as possible enlargement of such radius.

We may illustrate the Insertion Rule by Figure 9.1. Suppose we have 9 unequal circles in which 5 circles are "small". So after that we have found a configuration with the four larger circles by any algorithm like, e.g. MBH or PBH (see Figure 9.1(a)), next the insertion algorithm searches for a space in the given four-circle configuration by exploring the grid of points to insert the remaining largest circle (green color) in $\mathcal{S}_{\mathcal{R}}$. Once the algorithm finds a "free" space, it picks up the green colored (largest) circle from the set $\mathcal{S}_{\mathcal{R}}$ and places its center at the point of the grid where a "free" space has been detected (see Figure 9.1(b)). Notice in Figure 9.1(b) that the inserted circle partially overlaps with some of the other circles and also crosses the border of the container circle. After having detected a "free" space, the algorithm starts a local search from the newly created configuration in order to remove possible overlaps and reduce as much as possible the radius of the circular container. The algorithm is stopped when all free spaces have been tested. The new configuration with the added circle will be the one with the smallest radius of the circular container. In case during the search a configuration is detected with the same radius of the circular container as before the addition of the circle, then the algorithm stops returning this configuration. Once the new configuration is returned, the algorithm removes the added circle from $\mathcal{S}_{\mathcal{R}}$, selects the next largest circle in $\mathcal{S}_{\mathcal{R}}$ (the blue circle in the example), and repeats the above procedure until $\mathcal{S}_{\mathcal{R}}$ becomes empty. The pseudo-code structure of the insertion rule $\mathcal{I}_{\mathcal{R}}$ starting from an initial configuration X and trying to add circle s is as follows (τ denotes the local search procedure, while f returns the radius of the circular container for a given configuration):

$\mathcal{I}_{\mathcal{R}}(X, s)$
Step 1 (Init) Set $min_{rad} = +\infty$
Step 2 (Grid): create a regular grid T on the square region

containing the circular container
For each $(x_i, y_i) \in T$
If the space around (x_i, y_i) is "free" Then
Set $Y = X \cup \{(x_i, y_i)\}$
Set $X' = \tau(Y)$
If $f(X') < min_{rad}$ Then
Set $X^* = X'$
If $f(X') = f(X)$ Then
return X'
EndFor

Return X^*

Once we have defined the insertion procedure, we are ready to give a formal description of the whole algorithm:

Sequential Insertion Based MBH
Step 1(RemS): remove the set $\mathcal{S}_{\mathcal{R}}$ of all the "small" circles
Step 2(MBH): Apply MBH on the reduced problem.
Let X be the outcome of MBH
While $\mathcal{S}_{\mathcal{R}} \neq \emptyset$
Let $s \in \arg\max\{r_i : i \in \mathcal{S}_{\mathcal{R}}\}$
Set $X = \mathcal{I}_{\mathcal{R}}(X, s)$
Set $\mathcal{S}_{\mathcal{R}} = \mathcal{S}_{\mathcal{R}} \setminus \{s\}$
EndWhile
Return X

In the above algorithm MBH can be easily substituted by any other algorithm returning a configuration in the reduced space. In case MBH is replaced by PBH, the insertion procedure can be either applied to the best member of the final population, or, alternatively, to all members of the final population.

9.2 New perturbation moves

As already pointed out, when dealing with unequal circles, we can add new perturbation moves to the slight variant of the perturbation moves employed for equal circles. In particular, here we propose two further perturbation moves namely (i) the Random Jump (RJ) perturbation move and (ii) the Radius Based Random Swap (RBRS) perturbation move. The former could actually be employed also with equal circles (in fact, we will see that it is basically equivalent to the Jerk Perturbation move but less "local"). The latter can only be employed with unequal circles.

9.2.1 Random Jump (RJ) perturbation move

In Section 8.4 we have developed the Jerk Perturbation (JP) move technique in which circles' centers are perturbed within a neighbor space. The proposed Random Jump (RJ) perturbation move is actually quite similar: circles are

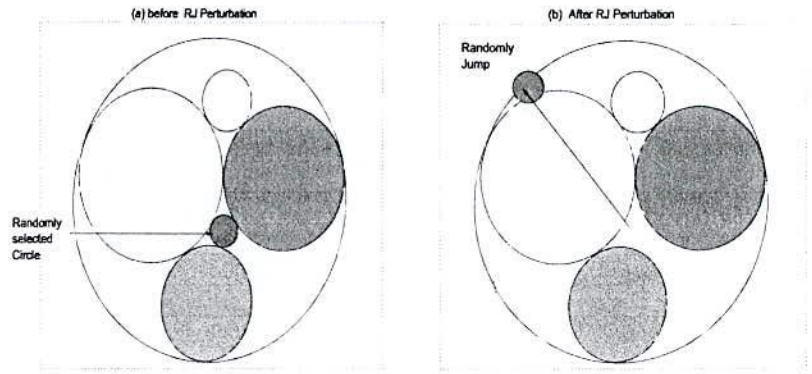

Fig. 9.2: Illustration of RJ perturbation move

randomly selected but rather than being slightly perturbed, they can jump within a large region (actually, in some sense the JP move can be regarded as a special case of the RJ perturbation move, in which only small jumps are allowed). Figure 9.2 illustrates the move. In Figure 9.2(a) we have a locally optimal configuration. In the example the RJ perturbation move randomly selects a single circle (the red one in the figure). Then, after the RJ perturbation the new configuration is displayed in Figure 9.2(b). Notice that the red circle jumps within a square region whose edge is $\sum r_i$ and is delimited by the dotted line in the figure, but crosses the border of the container and also overlaps with another circle. As usual, the local search procedure adjusts all the circles so that no overlaps occurs, and the circular container in such a way that its radius is as small as possible. The pseudo-code of the RJ perturbation moves is given below (the value of B_R, the diameter of the square boundary, is fixed to $\sum_{j=1}^{n} r_j$):

The pseudo-code of the RJ perturbation

Step 1: Let $Z = \{z_{11}, z_{12}, \ldots, z_{n1}, z_{n2}\}$ be a local minimum and set $Z' = Z$
Step 2: select $\Delta n \in (1, n)$ randomly
do i= 1 to Δn
Step 3: select $\hat{i} \in \{1, \ldots, n\}$ randomly
    Step 4: select $\Delta z_{\hat{i}k} \in (-B_R, B_R)$ randomly
  do k = 1 to 2
    Step 5: set $z'_{\hat{i}k} := z_{\hat{i}k} + \Delta z_{\hat{i}k}$
  End do
  Step 6: set $Z' = Z' \cap \left\{z'_{\hat{i}}\right\} / \{z_{\hat{i}}\}$
End do
return Z'

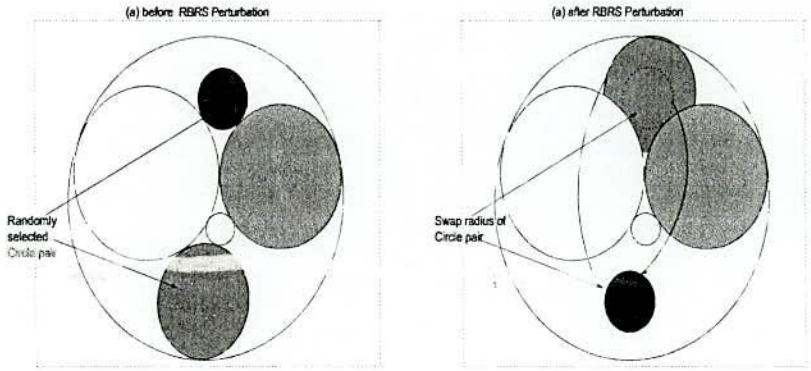

Fig. 9.3: Illustration of RBRS perturbation move

9.2.2 Radius Based Random Swap

In the Radius Based Random Swap (RBRS), first one or few pairs of circles are selected in such a way that in each pair the radii of the two circles are different (it will be soon clear that if two circles in a pair have the same radius, then the RBRS perturbation is meaningless). Then, we keep fixed the centers of the circles but swap their radii. For illustration, let Figure 9.3(a) represent a local optimal configuration; let the red and blue circles form the randomly selected pair. Then, after RBRS perturbation the new configuration is given in Figure 9.3(b). Notice that after swapping the radii, overlaps between circles occur and the red circle gets (slightly) outside the circular container. As usual, the local search procedure will adjust the situation in order to recover a feasible solution. The pseudo-code of the RBRS perturbation move is given below:

The pseudo-code of RBRS perturbation

Step 1: Let $Z = \{z_{11}, z_{12}, \ldots, z_{n1}, z_{n2}\}$ be a local minimum and set $Z' = Z$
Step 2: define $\Delta n \in (1, n/2)$ randomly/deterministically
Step 3: randomly select Δn distinct circles' pairs $(i_k, j_k), k = 1, \ldots, \Delta n$
so that $r_{i_k} \neq r_{j_k}$.
do k= 1 to Δn
Step 4: swap the two radii (r_{i_k}, r_{j_k})
End do
return Z'

9.3 Experiments and discussion

In this section we will perform some experiments to investigate different issues. In particular we will study:

- the performance of the proposed perturbations;
- the performance of the Sequential Insertion Based MBH (SIB-MBH) approach;

- the impact of the population.

Tab. 9.1: Test set with unequal circles

Test n.	n	Radii	BestKnown
1	6	$r_{1-3} = 10,\ r_{4-6} = 4.826$	21.5480
2	9	$r_{1-4} = 1,\ r_{5-9} = 0.41415$	2.4142
3	9	$r_{1-3} = 10,\ r_{4-9} = 3.533$	21.5470
4	12	$r_{1-3} = 10,\ r_{4-9} = 3.533,\ r_{10-12} = 2.3$	21.5470
5	13	$r_{1-3} = 10,\ r_{4-6} = 4.826,\ r_{7-12} = 2.371,\ r_{13} = 1.547$	21.5470
6	19	$r_{1-3} = 10,\ r_{4-6} = 4.826,\ r_{7-12} = 2.371,\ r_{13} = 1.547,$ $r_{14-19} = 1.345$	21.5470
7	19	$r_{1-3} = 10,\ r_{4-9} = 3.533,\ r_{10-12} = 2.3,\ r_{13-18} = 1.8,$ $r_{19} = 1.547$	21.5470
8	22	$r_{1-3} = 10,\ r_{4-6} = 4.826,\ r_{7-12} = 2.371,$ $r_{13} = 1.547,\ r_{14-19} = 1.345,\ r_{20-22} = 1.161,$	21.5470
9	25	$r_{1-3} = 10,\ r_{4-9} = 3.533,\ r_{10-12} = 2.3,\ r_{13-18} = 1.8$ $r_{19} = 1.547,\ r_{20-25} = 1.08$	21.5470
10	28	$r_{1-3} = 10,\ r_{4-6} = 4.826,\ r_{7-12} = 2.371,\ r_{13} = 1.547$ $r_{14-19} = 1.345,\ r_{20-22} = 1.161,\ r_{23-28} = 0.9$	21.5470
11	10	$r_1 = 50,\ r_2 = 40,\ r_{3-5} = 30,\ r_6 = 21$ $r_7 = 20,\ r_8 = 15,\ r_9 = 12,\ r_{10} = 10$	99.8850
12	11	$r_{1-2} = 25,\ r_{3-4} = 20,\ r_5 = 15,\ r_6 = 14$ $r_7 = 12,\ r_8 = 11,\ r_9 = 10.5,\ r_{10} = 10,\ r_{11} = 8.4$	60.8900
13	14	$r_1 = 40,\ r_2 = 38,\ r_3 = 37,\ r_4 = 36,\ r_5 = 35$ $r_6 = 31,\ r_7 = 27,\ r_8 = 23,\ r_9 = 19,\ r_{10} = 17,\ r_{11} = 16$ $r_{12} = 15,\ r_{13} = 14,\ r_{14} = 11$	114.9800
14	17	$r_1 = 25,\ r_2 = 20,\ r_{3-4} = 15,\ r_{5-7} = 10,\ r_{8-17} = 5$	49.6837
15	12	$r_{1-3} = 100,\ r_{4-6} = 48.26,\ r_{7-12} = 23.72$	215.4700
16	15	$r_1 = 1,\ r_{i+1} = r_i + 1,\ i = 1, \ldots, 14$	39.3700
17	17	$r_{1-4} = 100,\ r_{5-9} = 41.415,\ r_{10-17} = 20$	241.4214
18	162	$r_{1-3} = 1.8,\ r_4 = 1.75,\ r_{5-16} = 1.3,\ r_{17-25} = 1.05$ $r_{26-40} = 0.9,\ r_{41-71} = 0.8,\ r_{72} = 0.75,\ r_{73-83} = 0.7$ $r_{84-137} = 0.65,\ r_{138-162} = 0.55$	11.7300

The test instances which will be considered are those reported in [106]. These are 18 test instances for the case of unequal circles.[1] The characteristics of each test are indicated in Table 9.1. In such table column `Test n.` denotes the identifier of the instance; column n denotes the number of circles of the instance; column `Radii` denotes the different radii of the circles in the instance; column `BestKnown` denotes the best known value in the literature for the instance.

9.3.1 *Experiments with different perturbation moves and with the sequential insertion strategy*

The different perturbation moves which will be tested are the Full Jerk (FJ) one (with perturbation range $\Delta_i = 0.8r_i$ for the coordinates of the i-th circle), the Random Jump (RJ) perturbation move (with a number of randomly selected

[1] Actually, in the paper [106] 24 test problems are reported, but four of them are with equal circles (namely, tests n.2, 21-23), and two of them are equivalent to other problems within the test set (namely, test n.7 is equivalent to test n.19, and test n.13 is equivalent to test n.20).

Tab. 9.2: The Impact of Removal Strategy in Sequential insertion based approach

Test n.	n	Reduced circles
1	6	6
2	9	9
3	9	9
4	12	9
5	13	6
6	19	6
7	19	9
8	22	6
9	25	9
10	28	6
11	10	8
12	11	11
13	14	14
14	17	7
15	12	6
16	15	12
17	17	9
18	162	162

Tab. 9.3: The performance of MBH approaches with different perturbation moves and insertion strategies. Note that in this table OurBestResults is denoted as OBRs.

Test n.	OBRs	No. of success (Result) of MBH approach			No. of success (Result) of SIB-MBH approach		
		FJ	RJ	RBRS	FJ	RJ	RBRS
1	21.5480	50	15	50	50	15	50
2	2.4142	12	15	37	12	15	37
3	21.5470	14	10	40	14	10	40
4	21.5470	8	1	19	14	10	40
5	21.5480	3	0	5	3	2	9
6	21.5470	0	0	0	19	14	50
7	21.5470	0	0	0	14	10	40
8	21.5480	0	0	0	20	16	50
9	21.5470	0	0	0	14	15	38
10	21.5470	0	0	0	20	15	24
11	99.8850	0	0	12	0	0	50
12	**60.7099**	0	0	12 **(60.7099)**	0	0	12 **(60.7099)**
13	**113.5552** (♠)	10 (114.0814)	0	6 (113.5846)	10 114.0814)	0	6 (113.5846)
14	**49.1873**	1 (49.31945)	10 (49.6498)	1 (49.2470)	6 **(49.1873)**	100 **(49.1873)**	50 **(49.1873)**
15	215.4700	5	0	7	31	16	50
16	**38.8380**	1 (38.9189)	10 (39.2962)	1 **38.8380**	0	3 (39.3534)	5 **(38.8380)**
17	241.4214	0	0	7	7	8	28
18	**11.5119** (♣)	1 (11.5336)	5 (11.6599)	1 (11.5422)	1 (11.5336)	5 (11.6599)	1 (11.5422)
Failure	0	8	11	5	3	3	0
Success	18	10	7	13	15	15	18
Imp.	5	4	3	5	3	3	5
B. Imp.	5	0	0	2	1	1	3

circles, on which the perturbation is carried on, equal to $\lceil n/20+1 \rceil$), and the Radius Based Random Swap (RBRS) one (with $\lceil n/20+1 \rceil$ randomly selected pairs of circles on which the perturbation is carried on). Experiments are performed both with the standard MBH approach and with the sequential insertion strategy, i.e. with the SIB-MBH approach. In all cases we set `MaxNonImp` =200. The number of runs is $R = 50$ for all tests except for the highly computationally demanding Test n. 18 for which we reduced the number of runs to $R = 6$.

For what concerns SIB-MBH, we report in Table 9.2 for each test instance the total number n of circles for the instance and the reduced number of circles after removal of the "small" circles. We observe from the table that for some test instances like, e.g., n. 6-10 a large number of "small" circles is removed. For some other instances a lower number of circles is removed (like, e.g., instances n. 4 and 11). Finally, for the test instances n. 1, 2, 3, 12, 13 and 18 there is no "small" circle and, consequently, the MBH and SIB-MBH approach are equivalent ones.

The results are reported in Table 9.3. Column `Test n.` denotes the identifier of the test instance as indicated in Table 9.1. Column `OBRs` (OurBestResult) denotes the best results we could obtain during all our experiments. Note that in all cases a value (in the Column `OBRs`) at least as good as the `BestKnown` one in the literature as reported in Table 9.1, is reached and that values in **boldface** indicate better results compared to the `BestKnown` ones. The next following three columns report the number of successes on each instance for each of the three perturbation moves tested (FJ, RJ, RBRS) with MBH approach. The last three columns report the number of successes on each instance for each of the three perturbation moves tested (FJ, RJ, RBRS) with SIB-MBH, i.e. with the use of the sequential insertion strategy. It is worthwhile to explain here what do we mean by number of successes. When the number of successes is equal to 0, this means that the approach was unable to reach the best known result in the literature. For all the instances for which the best result obtained by an approach was at least as good as the best known one in the literature, the number of successes is the number of runs where the best result has been obtained. In the latter case, when a result better than the best known one in the literature could be obtained, we also report within parenthesis such result. We also remark here that for Test n.13, we have obtained the best result – **113.5552**$^{\spadesuit}$ by the 500 runs of SIB-MBH(RBRS) approach [2], while for Test n.18 we have obtained the best result– **11.5119**$^{\clubsuit}$ by the SIB-PBH(FJ) approach discussed later on.

In the table the row named `Failure` reports the total number of instances where the approach was unable to reach the best known result in the literature (or, equivalently, the number of instances for which the number of successes is equal to 0). Similarly, row `Success` reports the total number of instances where the approach was able to obtain a solution at least as good as the best known one. Row `Imp.` reports the total number of instances for which the approach was able to obtain an improved solution and, finally, row `B. Imp.` indicates the total number of instances for which the approach was able to obtain an improved solution which is also the overall best among all those obtained in the

[2] Such experiment is not discussed here but can be found in [96].

different experiments.

Now we briefly discuss the results reported in Table 9.3. At first we consider the performance of the different perturbations moves within the standard MBH approach, i.e. MBH without sequential insertion strategy. We observe that the MBH(RBRS) approach was able to obtain success in 13 instances out of 18; in five instances it was able to improve the available `BestKnown` value and in three cases the improvement is a best one. Unfortunately, there are also five failures. Note that enlarging the number of runs only partially helps. Indeed, when we extended the number of runs to $R = 500$, we could get at least one success for all the instances but still two failures, namely for the two tests n. 9 and 10. The situation is even worse for the MBH(FJ) and MBH(RJ) approaches, for which the number of failures is clearly higher compared to that of the MBH (RBRS) approach; moreover, though there are some improvements in both the approaches, none of them has a best improvement.

Things get definitely better when we consider the performance of the different perturbations moves in the SIB-MBH approach. The SIB-MBH(RBRS) approach has no failure, five improvements and three best improvements; both SIB-MBH(FJ) and SIB-MBH(RJ) approaches, though inferior with respect to SIB-MBH(RBRS), have only three failures but one best improvement. If we focus our attention on the comparison between MBH(RSBS) and SIB-MBH(RSBS), we can remark that the five failures in MBH(RSBS) occur with instances n. 6-10, for which the SIB-MBH(RSBS) approach first removes a relatively large number of ("small ") circles. Once such circles are removed, the problems get quite easy ones and the following sequential insertion can always be carried on relatively easily without having to enlarge the radius of the circular container (i.e., all the missing circles can be inserted in the "holes" of the container). This is not always the case. Instance n. 14 deserves some attention. The different runs of MBH over the reduced space return two distinct solutions with the seven remaining circles, one with radius 48.6111 and the other with radius 48.922, so that the first one is clearly better than the second one. But when moving to the second phase (sequential insertion of the missing circles), the situation is reversed: the first solution leads to a solution with radius 49.2296, while the second one leads to a better solution with radius 49.1873. Basically, the second solution has a worse radius but larger holes where the missing circles can be placed. Therefore, what we can conclude from this is that it is often a good strategy to perform the insertion of missing circles not only from the best solution returned by the first phase, but also from some suboptimal solutions obtained during the first phase, because the latter may lead to better solutions after insertion of the missing circles.

The final indications of this set of experiments are quite clear:

- the use of the sequential strategy clearly enhances the performance of all the approaches, independently from the perturbation move employed (for a given perturbation move the performance with the sequential strategy is almost always better than the one without);
- the RBRS move is a clear winner with respect to the FJ and RJ moves with a lower number of failures, a higher number of improvements and best

Tab. 9.4: The total elapsed CPU times of the experiments for SIB-MBH(RBRS) approach

Test n.	Elapsed time (sec)
1	39
2	75
3	104
4	440
5	35806
6	17607
7	3778
8	11768
9	63890
10	52967
11	213
12	264
13	468
14	4975
15	813
16	249
17	3130
18	407137

improvements, and with a number of successes almost always larger than those obtained with the other moves. The only exception is represented by instance n.18. The peculiarities of this instance and a possible explanation for the worse behavior of RBRS with respect to FJ on it, will be discussed later on.

Some attention should be focused on instance n.13 and, even more, on instance n.16. In both cases all radii are different and the different between two consecutive radii is relatively small. In these cases the MBH(RBRS) approach (or the SIB-MBH(RBRS) approach) works much better than the approaches with other perturbation moves. For instance n.16 we observed a large variability of the final solutions and, in spite of the relatively small dimension, this instance turns out to be particularly challenging. We remark that instance n.16 is one (actually of moderate size) among those proposed in the Circle Packing Contest (see `http://www.recmath.org/contest/CirclePacking/index.php`), and its difficulty seems to confirm that such instances are more challenging than the other test instances with unequal circles reported in the literature. More generally, our impression is that the hardest instances for the case of unequal circles are those with many circles with slightly different radii. For a discussion about how to deal with the instances of the contest we refer to [3].

As a final comment, we emphasize once again that the proposed approaches and, in particular, the SIB-MBH(RBRS) one, turn out to be extremely efficient when compared with the existing literature, being able to get at least the same results and, in some cases, also to considerably improve the best known results as reported in [106] (and in Table 9.1). We also report in Table 9.4 the overall computation time required for $R = 50$ runs on all the test instances except for the instance n.18, for which $R = 6$, as already mentioned.

Tab. 9.5: Impact of Population in RBRS perturbation moves on sequential insertion based approach. Note that in this table OurBestResults is denoted as OBRs.

Test n.	OBRs	No. of success in % (result)					
		$N_p = 1$	$N_p = 2$	$N_p = 4$	$N_p = 5$	$N_p = 8$	$N_p = 10$
1	21.5480	100	100	100	100	100	100
2	2.4142	74	100	100	100	100	100
3	21.5470	80	100	100	100	100	100
4	21.5470	80	100	100	100	100	100
5	21.5480	18	100	100	100	100	100
6	21.5470	100	100	100	100	100	100
7	21.5470	80	100	100	100	100	100
8	21.5480	100	100	100	100	100	100
9	21.5470	76	100	100	100	100	100
10	21.5470	48	100	100	100	100	100
11	99.8850	100	100	100	100	100	100
12	**60.7099**	**24(60.7099)**	**44(60.7099)**	**67(60.7099)**	**80(60.7099)**	**33(60.7099)**	**80(60.7099)**
13	**113.5552**♠	12(113.5846)	12(113.7753)	16(113.8376)	40(113.9434)	33(114.02999)	100(113.59499)
14	**49.1873**	**100(49.1873)**	**96(49.1873)**	**100(49.1873)**	**100(49.1873)**	**100(49.1873)**	**100(49.1873)**
15	215.4700	100	100	16	100	100	100
16	**38.8380**	**10(38.8380)**	**28(38.8380)**	**42(38.8380)**	**70(38.8380)**	**50(38.8380)**	**80(38.8380)**
17	241.4214	56	100	100	100	100	100
18	**11.5119**♣	12 (11.5422)	8(11.5256)	50(11.5410)	30(11.5410)	100(11.5416)	60(11.5369)
100 % Suc.		6	13	13	15	15	15
50% <Suc. < 100%		6	1	1	0	0	3
5% <Suc.< 50 %		6	4	4	3	3	0
Best Impro.		3	3	3	3	3	3

9.3.2 *Impact of population*

In this section we investigate the PBH approach with different population sizes. Following the indications obtained from the previous set of experiments, we will first restrict our attention to SIB-PBH(RBRS) (sequential insertion will be performed starting from all the members of the final population).

We consider the population sizes $N_p = 2, 4, 5, 8, 10$ as well as $N_p = 1$ (i.e. the SIB-MBH(RBRS) approach). In Table 9.5 we report the results in terms of percentage of successes obtained with: 50 runs of SIB-MBH(RBRS), 25 runs of SIB-PBH(RBRS) with $N_p = 2$, 12 runs of SIB-PBH(RBRS) with $N_p = 4$, 10 runs of SIB-PBH(RBRS) with $N_p = 5$, 6 runs of SIB-PBH(RBRS) with $N_p = 8$, and 5 runs of SIB-PBH(RBRS) with $N_p = 10$. This way the overall computational effort with the different population sizes is approximately the same. We set MaxNonImp =200 for all the population sizes.

The distance dissimilarity measure in SIB-PBH(RBRS) approach is similar to (7.1), the one employed with equal circles, but with a slight difference. Given a local minimizer X, in vector δ_X we first place the distances with respect to the barycenter of the circles with largest radius, ordered in a nondecreasing way, then the distances with respect to the barycenter of the circles with second largest radius, ordered again in a nondecreasing way, and so on for all the different radii. Then, we define the dissimilarity measure as follows:

$$\mathcal{D}(X, Y) = \sum_{k=1}^{n} |\delta_X[k] - \delta_Y[k]|.$$

We observe in the table that both MBH as well as PBH based approaches are able to obtain at least the best known value in the literature (no failure is observed) , and in some cases, to improve it. Even MBH turns out to be able to reach good percentage of successes in most instances. On the other hand, as we increase the population size, the robustness of the method also increases, reaching 100% in almost all instances (15 out of 18) for $N_p \geq 5$.

Tab. 9.6: The impact of FJ and RBRS perturbation moves on sequential insertion based approaches in presents of population. Note that in this table OurBestResults is denoted as OBRs.

Test n	BestKnown (B.K.)	OBRs	Results for $N_p = 5$ with $R = 10$		Results for $N_p = 10$ with $R = 5$	
			FJ	RBRS	FJ	RBRS
12	60.8900	**60.7099**	61.8213(1)	**60.7099** (8)	61.3821(1)	**60.7099** (4)
13	114.9800	**113.5552**	115.8722(1)	113.9434(4)	115.7018(1)	113.59499(5)
14	49.6837	**49.1873**	**49.1873** (1)	**49.1873** (10)	**49.1873** (1)	**49.1873** (5)
16	39.3700	**38.8380**	39.4056(1)	**38.8380**(7)	39.4980(1)	**38.8380**(4)
18	11.7300	**11.5119**	11.5242(1)	11.5410(3)	**11.5119** (1)	11.5369(3)
Total Failure		0	3	0	3	0
Best Improve		5	1	3	2	3

9.3.3 *Comparison of PBHs with different perturbation moves*

We decided to perform also some experiments to compare the performance of PBH (actually, SIB-PBH) with the FJ perturbation move as well as with the RBRS perturbation move. We only considered $N_p = 5, 10$ with the number of runs $R = 10, 5$ respectively. We also restricted the instances to the five ones for which we were able to improve the best known results in the literature, i.e. Tests n. 12, 13, 14, 16 and 18. The experimental results are reported in Table 9.6. It can be clearly seen that in all cases, except instance n.18, the FJ perturbation move, which does not take into account the combinatorial nature of the problem, delivers results inferior to those obtained with the RBRS perturbation move based on swapping the centers of circles with different radii.

Test n.18 deserves a separate comment. For this case it seems that the FJ perturbation move is better than the RBRS one (this was also observed in Table 9.3). If we look at this instance, we notice that it contains a large number of circles with the same radius (e.g., 54 circles, one third of the total number of circles, have radius equal to 0.65). It is possible that such circles occupy a portion of the container which can not be optimized by swapping moves (recall that such moves only involve circles with different radii), while it can be optimized efficiently by random perturbations. Seen in another way, we have two distinct aspect in a problem with unequal circles: a *continuous* one, represented by the fact that circle centers have to be chosen in $\mathbb{R}^2$, and a *combinatorial* one, due to the different radii of the circles. In the case of circles with all equal radius the combinatorial component simply does not exist, while in case there are a lot of (or even all, as in instances n.13 and 16) circles with different radii, the combinatorial component is more relevant than the continuous one. In case of test n.18, with few different radii and many circles with the same radius, it seems that taking into account the continuous aspect (through the use of the random FJ perturbation) is more important than taking into account the combinatorial aspect (through the use of swapping moves). Something which could be explored in the future is a mixed strategy, where both swap moves and random ones are employed.

10. CONCLUSION AND FUTURE RESEARCH

In the thesis we have dealt with two optimization problems, maximin Latin Hypercube Design (LHD) and packing equal and unequal circles into a circular container. Such two classes of optimization problems differ for the nature of their feasible region: in the former the feasible region is a discrete set (the problem is a combinatorial optimization one), while in the latter the feasible region is continuous. Moreover, such problems usually arise in different contexts and applications. However, in spite of these differences, there are many similarities between them: in both we have to place "objects" (respectively, design points and circles) in a region (respectively, a hypercube and a circular container) in such a way that some constraints are satisfied (respectively, no common coordinate of the design points and no overlapping between circles' interiors) and some quality measure is optimized (respectively, the minimum distance between the points to be maximized, and the radius of the circular container, to be minimized). The similarities are even stronger between maximin LHD and the problem of packing equal circles. Indeed, the latter is equaivalent to the problem of placing points within a circular container with fixed radius in such a way that their minimum distance is maximized, exactly like in maximin LHD.

The similarities between the problems suggested to study similar heuristic approaches for them. Maximin LHD has been attacked with Iterated Local Search (ILS) heuristics, while packing problems have been attacked with Monotonic Basin Hopping (MBH) heuristics and their population-based variant PBH. In spite of the different names (used to follow the current terminology in the literature) the two approaches are quite similar and, in fact, MBH can be viewed as the continuous counterpart of ILS (typically used in the context of combinatorial optimization problems).

Below we summarize the main findings of the thesis and discuss about possible future research directions.

10.1 Contributions

In the thesis we have proposed and computationally tested different variants of ILS and MBH approaches respectively for the maximin LHD and packing problems. The proposed algorithms achieve a breakthrough to obtain optimal solutions compared with existing methods. In the following subsections we discuss about such achievements.

10.1.1 Maximin LHD

In the definition of ILS approaches for maximin LHD a major role has been played by the definition of the perturbation move operator. We have proposed mainly two Perturbation Moves (PM) named Cyclic Order Exchange (COE) and Pairwise Crossover (PC) and their variants. Both PM operators turn out to be effective and efficient. COE is better when (N, k) are small, while PC is better when (N, k) are relatively large. A quite relevant finding is that higher quality results can be obtained if the most natural optimality criterion $\text{Opt}(D_1, J_1)$, which only takes into account the minimum distance D_1 in a LHD and the number J_1 of its occurences, is substituted by the optimality criterion $\text{Opt}(D_1, \phi)$, where LHDs are compared through function ϕ, which takes into account all distances in a LHD (with a decreasing weight as the distance gets larger). In our opinion, a remarkable finding of the thesis is that a monotonic search with respect to ϕ values but *non monotonic* with respect to the D_1 values, returns (much) better results with respect to a simple monotonic search with respect to the (D_1, J_1) values. It seems that the search through the ϕ values allows effective backtracking moves with respect to the (D_1, J_1) values.

We have also discussed about the time complexity of the ILS algorithms. It is not possible to give an exact time complexity of the ILS approaches but, mixing theoretical considerations and computational experiments, we have derived empirical formulas returning computation times as a function of the N and k values.

Finally we have compared the proposed ILS approaches with the existing literature. Experimentally it is shown that such approaches are competitive with existing ones and in many cases they outperform them. The algorithms have been able to obtain a large number of improved maximin LHD values which have been recently uploaded in the web site `http://www.spacefillingdesigns.nl/` (see [276]).

Though the proposed algorithms have been able to come state-of-the-art, there is still place for improvements. In particular, for small k values it has been observed that a different approach, Periodic Design (PD), is still able to outperform the proposed ILS approaches for sufficiently large N values.

10.1.2 Packing problems

For these problems we discussed the Monotonic and Population Basin Hopping approaches. Their performance turned out to be quite good (with many improvements with respect to the existing literature). But besides deriving such results, our aim were that of analyzing the each components of the approaches in order to study their impact and to choose carefully their definition. In particular, the experiments revealed that:

- local searches alone are not enough: the simple Multistart approach, where local searches are started from randomly sampled points, performs much worse than a carefully designed MBH approach;
- in the case of equal circles, the "optimal" size Δ of the perturbation where

circles are randomly shifted within a region whose size is controlled by Δ, is inversely proportional with respect to the number of perturbed circles; moreover, it also appears that an intermediate choice between perturbing only few circles or all of them is the best option;

- in the case of unequal circles, the better choice between a random shift of the circles like in the case of equal circles, and a combinatorial move where radii of the circles are swapped, depends on the ratio between the number of different radii's values and the total number of circles: as the ratio increases, combinatorial moves become preferable;
- the experiments show that there is not an optimal choice for the stopping parameter `MaxNonImp` (sometimes even for small n values, like $n = 31$, a longer search is better) but they identify robust choices for it;
- in the case of unequal circles, removal of small circles is often essential to solve the problems, but it is important to observe that there is not a monotonic relation between partial and complete configurations, i.e. while a partial configuration has a better radius than another one, the situation can be reversed when adding missing circles;
- on average, the best performance is obtained with MBH or with PBH with a small population, but PBH with a larger population seems to guarantee more robustness, with good results also over the instances where MBH has to struggle.

10.2 Future Research

Though extensive experiments have been performed for a careful choice of the algorithms' components and of the parameter values, and for a comparison with the existing literature, we believe that a major issue for the future is a further analysis, not merely from the experimental point of view but also from the theoretical one, of the algorithms as well as of the problems at hand. Exploiting theoretical properties of the problems could allow, e.g., to reduce the search space and improve the quality of the results.

Moreover other possible directions for future works could include:

- testing the proposed ILS approaches with other optimality criteria;
- investigate and improve the Local Search as well as Perturbation Move operators;
- improve the code to reduce time complexity;
- extend the approaches to other packing problems;
- develop more robust perturbations moves;
- develop more robust dissimilarity measures.

BIBLIOGRAPHY

[1] Addis B., M. Locatelli, and F. Schoen, *Packing circles in a square: new putative optima obtained via global optimization*, Optimization Online (2005). Available in http://www.optimization-online.org/DB HTML/2005/03/1088.html.

[2] Addis B., M. Locatelli, and F. Schoen, *Local optima smoothing for global optimization*, Optimization Methods and Software, vol. 20 (4-5), pp. 417437(2005).

[3] Addis B., M. Locatelli, and F. Schoen, *Efficiently packing unequal disks in a circle*, Operations Research Letters, vol. 36 (1), pp. 37-42 (2007).

[4] Addis B., M. Locatelli , and F. Schoen, *Disk packing in a square: a new global optimization approach*, (2008). to appear in INFORMS Journal on Computing .

[5] Akiyama J., R. Mochizuki, N. Mutoh, and G. Nakamura, *Maximin Distance for n Points in a Unit Square or a Unit Circle*, Discrete and Computational Geometry, Springer Berlin / Heidelberg, vol. 2866, pp. 9-13 (2003).

[6] Alam, F. M., K. R. McNaught, and T. J. Ringrose, *A comparison of experimental designs in the development of a neural network simulation metamodel*, Simulation Modeling: Practice and Theory, vol. 12 (7-8), pp. 559-578(2004).

[7] Applegate D., W. Cook, and A. Rohe, *Chained Lin-Kernighan for large traveling salesman problems*, Technical Report No. 99887, Forschungsinstitut für Diskrete Mathematik, University of Bonn, Germany,(1999).

[8] Aarts E. H. L., and J. K. Lenstra, Local search in combinatorial optimization, *Wiley-Interscience Series in Discrete Mathematics and Optimization*, Chichester, John Wiley & Sons,(1997).

[9] Audze P., and V. Eglais, *New approach to planning out of experiments*, problems of dynamics and strength, vol. 35, pp. 104107(1977). (in Russian).

[10] Baer D., *Punktverteilungen in Würfeln beliebiger Dimension bezüglich der Maximum-norm*, Wiss. Z. Pädagog. Hochsch. Erfurt/Mühlhausen, Math.-Naturwiss, Reihe, vol. 28, pp. 87-92(1992).

[11] Bahl P., and V. N. Padmanabhan, *RADAR: An In-Building RF-Based User Location and Tracking System*, Proceedings of IEEE INFOCOM 2000, Tel-Aviv, Israel, vol. 2, pp. 775784 (March 2000).

[12] Barvinok A., *Problems of Distance Geometry and Convex Properties of Quadratic Maps. Disc*, Comp. Geom., vol. 13, pp.189-202(1995).

[13] Barthelemy J. F. M., and R. T. Haftka, *Approximation concepts for optimum structural design – A review*, Structural Optimization, vol. 5(3), pp. 129-144(1993).

[14] Bateman, P., and P. Erdös, *Geometrical extrema suggested by a lemma of Besicovitch*, Amer. Math. Monthly, vol. 58, pp. 300314(1951).

[15] Bates R. A., R. J. Buck, E. Riccomagno, and H. P. Wynn, *Experimental design and observation for large systems*, Journal of the Royal Statistical Society: Series B, vol. 58, pp. 77-94(1996).

[16] Bates S. J., J. Sienz, and V. V. Toropov, *Formulation of the optimal Latin hypercube design of experiments using a permutation genetic algorithm*, AIAA 2004-2011, pp. 1-7(2004).

[17] Battiti R., and M. Protasi, *Reactive search, a history-based heuristic for the MAX-SAT*, ACM Journal of Experimental Algorithmic, vol. 2, (1997).

[18] Baum E. B., *Toward practical neural computation for combinatorial optimization problems*, Neural networks for computing (ed: Denker J.), Boston: American Institute of Physics, pp. 5364 (1986).

[19] Baum E. B., *Toward practical 'neural'computation for combinatorial optimization problems*, Neural Networks for Computing (Ed.: Denker J.), AIP conference proceedings, pp. 5364 (1986).

[20] Baum E. B., *Iterated descent: A better algorithm for local search in combinatorial optimization problems*, Technical report, Caltech, Pasadena, CA, (1986).

[21] Baxter J., *Local optima avoidance in depot location*, Journal of the Operational Research Society, vol. 32, pp. 815819 (1981).

[22] Berman H. M., J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, *The Protein data bank*, Nucleic Acids Research, vol. 28, pp. 235-242(2000).

[23] Bettonvil B., and J. P. C. Kleijnen, *Searching for Important Variables in Simulation Models with Many Variables: Sequential Bifurcation,* European Journal of Operations Research, vol. 96, pp. 180-194, 1996.

[24] Bible S. R., M. Zyda, and D. Brutzman, *Using spread-spectrum ranging techniques for position tracking in a virtual environment*, In Second IEEE Workshop on Networked Realities, Boston, MA, (October 1995).

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

[26] Blum C., and A. Roli, *Metaheuristics in combinatorial optimization: Overview and conceptual comparison*, ACM Computing Surveys, vol. 35(3), pp. 268308(2003).

[27] Boll D. V., J. Donovan, R. L. Graham, and B. D. Lubachevsky , *Improving dense packings of equal disks in a square*, The Electronic J. of Comb., vol. 7, R46, (2000).

[28] Booker A. J., J. E. Dennis, P. D. Frank, D. B. Serafini, V. Torczon, and M. W. Trosset, *A rigorous framework for optimization of expensive functions by surrogates*, Structural Optimization, vol. 17, pp. 1-13(1999).

[29] Boyd S., P. Diaconis, and L. Xiao, *Fastest mixing Markov chain on a graph*, SIAM Review, vol. 46(4), pp. 667-689(2004).

[30] Büadoiu M., *Approximation Algorithm for Embedding Metrics into a Two-Dimensional Space*, Proc. 14th SODA, pp. 434-443(2003).

[31] Büadoiu M., E. D. Demaine, M. T. Hajiaghayi, P. Indyk, *Low-Dimensional Embedding with Extra Information*, Proc. 20th SoCG, pp. 320-329(2004).

[32] Bursztyn D., and D. M. Steinberg, *Comparison of designs for computer experiments*, Journal of Statistical Planning and Inference, vol. 136(3), pp. 1103-1119 (2006).

[33] Butler N. A., *Optimal and orthogonal Latin hypercube designs for computer experiments*, Biometrika, vol. 88(3), pp. 847-857(2001).

[34] Casado, L. G., I. García, P .G. Szabó, and T. Csendes, *Packing Equal Circles in a Square II. - New results for up to 100 circles using the TAMSASS-PECS stochastic algorithm*, Optimization Theory: Recent Developments from Mátraháza (Eds: Giannessi F., P. Pardalos, T. Rapcsak), Kluwer Academic Publishers, Dordrecht, Boston, London, pp. 207-224 (2001).

[35] Castillo I., F. J. Kampas, and J. D. Pinter, *Solving circle packing problems by global optimization: Numerical results and industrial applications.* To appear in European Journal of Operational Research.

[36] Cha A., *Electronic tracking is finding new uses*, San Jose Mercury News, Sunday, Jan. 16, (2005).

[37] Chaloner, K., and I. Verdinelli, *Bayesian Experimental Design: A Review*, Technical Report # 599, Department of Statistics, Carnegie Mellon University, (1994).

[38] Chatterjee, K., L. Deng, and D. Lin, *Resolution V.2 Search Designs*, Communications in Statistics: Theory and Methods, Vol. 29(5-6), pp. 1143-1154 (2000).

[39] Chen C. S., B. Ram, and S. Sarin, *An integer programming model for a class of assortment problems*, Report – Dept. of Ind. North Carina. A & T State university, Greensboro,(1989).

[40] Cioppa, T. M., Efficient nearly orthogonal and space-filling experimental designs for high-dimensional complex models, *PhD thesis*, naval postgraduate school Monterey, California,USA,(2002).

[41] Clare B. W., and D. L. Kepert, *The closest packing of equal circles on a sphere*, Proc. Roy. Soc. London Ser. A 405, pp. 329-355(1986).

[42] Correia M. H. , J. F. Oliveira, and J. S. Ferreira, *Cylinder packing by simulated annealing*, Pesquisa Operacional, vol. 20 (2), pp. 269286 (2000).

[43] Coxeter H. S. M., *The Problem of Packing a Number of Equal Non overlapping Circles on a Sphere*, Trans. New York Acad. Sci., vol. 24, pp. 320-331(1962).

[44] Crary S. B., *Design of computer experiments for metamodel generation*, Analog Integrated Circuits and Signal Processing, vol. 32(1), pp. 7 - 16(2002).

[45] Crary S. B., P. Cousseau, D. Armstrong, D. M. Woodcock, E. H. Mok, O. Dubochet, P. Lerch, and P. Renaud, *Optimal design of computer experiments for metamodel generation using I-OPT^{TM}*, Computer Modeling in Engineering & Sciences, vol. 1(1), pp. 127-139(2000).

[46] Croft H. T., K. J. Falconer, and R. K. Guy, Unsolved Problems in Geometry, *Springer, New York*, (1991).

[47] Currin C., Mitchell, T., Morris, M. D., and D. Ylvisaker, *Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments*, Journal of the American Statistical Association, vol. 86, pp. 953-963 (1991).

[48] Dam E. R. van, *Two-dimensional minimax Latin hypercube designs*, CentER Discussion Paper 2005-105. Tilburg University,(2005).

[49] Dam E. R. van, B. G. M. Husslage, D. den Hertog, and J. B. M. Melissen, *Maximin Latin hypercube designs in two dimensions*, Operations Research, vol. 55(1), pp. 158169(2007).

[50] Dam E. R. van, G. Rennen, and B. Husslage, *Bounds for maximin Latin hypercube designs*, Department of Econometrics and Operations Research, Diss. Paper, Tilburg University, The Netherlands,(2007).

[51] Darrell T., G. Gordon, M. Harville, and J. Woodfill, *Integrated person tracking using stereo, color, and pattern detection*, In Conference on Computer Vision and Pattern Recognition, IEEE Computer Society, pp. 601-608 (June 1998).

[52] Dimnaku A., R. Kincaid, and M. W. Trosset, *Approximate solutions of continuous dispersion problems*, 2002 ISOLDE conference, http://www.math.wm.edu/ ˜trosset/,(2002).

[53] Dong Q., and W. Zhijun, *A geometric build-up algorithm for solving the molecular distance geometry problem with sparse distance data*, J. of Global Optimization, vol. 26(3), pp. 321-333 (2003).

[54] Dorfman, R., *The Detection of Defective Numbers of Large Populations*, The Annals of Mathematical Statistics, vol. 14, pp. 436-440 (1943).

[55] Dorit S., Hochbaum, and W. Maass, *Approximation schemes for covering and packing problems in image processing and VLSI*, Journal of the Association for Computing Machinery, vol. 1(32), pp. 130136 (1985).

[56] Dowsland K. A., *Palletisation of cylinders in cases*, Operation research Spectrum, vol. 13, pp. 171172 (1991).

[57] Dowsland K. A., and W. B. Dowsland, *Packing problems*, European Journal of Operational Research, vol. 56 (1), pp. 214 (1992).

[58] Dowsland K. A., M. Gilbert, and G. Kendall, *A local search approach to a circle cutting problem arising in the motor cycle industry*, Journal of the Operational Research Society, Palgrave Macmillan, vol 58(4), pp. 429-438(2007).

[59] Driessen L. T., H. P. Stehouwer, and J. J. Wijker, *Structural mass optimization of the engine frame of the Ariane 5 ESC-B*, Proceedings of the European Conference on Spacecraft, Structures, Materials & Mechanical Testing, Toulouse, France, (2002).

[60] Dyckhoff H., *A typology of cutting and packing problems*, Eur. J. Oper. Res, vol. 44, pp. 145159 (1990).

[61] Dyckhoff H., G. Scheithauer, and J. Terno, *Cutting and packing (C & P)*, Annotated Bibliography in Combinatorial Optimization (Eds: DellAmico M., F. Maffioli, S Martello), Wiley, Chichester, pp. 393413(1997).

[62] Epstein L., and R. van Stee. *On variable-sized multidimensional packing*, In Algorithms - ESA 2004, Proceedings Twelfth Annual European Symposium, Lecture Notes in Computer Science, Springer, vol. 3221, pp. 287-298 (2004).

[63] Erkut E., *The discrete para-dispersion problem*, European Journal of Operational Research, vol. 46(1), pp. 48-60 (1990).

[64] Fang K. T., and Y. Wang, Number-Theoretic Methods in Statistics, *Chapman and Hall*, London, (1994).

[65] Fang K. T., C. Ma, and P. Winker, *Centered L2-Discrepancy of Random Sampling and Latin Hypercube Design, and Construction of Uniform Designs*, Mathematics of Computation, vol. 71(237), pp. 275-296 (2000).

[66] Fang K. T., D. K. J. Lin, P. Winker, and Y. Zhang, *Uniform Design: Theory and Application*, Technometrics, vol. 42(3), pp. 237-248 (2000).

[67] Fang K. T., R. Li, and A. Sudjianto, Design and Modeling for Computer Experiments, *CRC Press*, New York, (2006).

[68] Flourney N., *A Clinical Experiment in Bone Marrow Transplantation: Estimating a Percentage Point of a Quantal Response Curve In Case Studies in Bayesian Statistics*, Springer-Verlag(Eds: C. Gatsonis et al.), New York, pp. 324-336 (1993).

[69] Fodor F., *The Densest Packing of 19 Congruent Circles in a Circle*, Geometriae Dedicata, Springer Netherlands, Springer Link, vol. 74(2), (2004).

[70] Fodor F., *The Densest Packing of 12 Congruent Circles in a Circle*, Contributions to Algebra and Geometry(Beitrage zur Algebra und Geometrie), vol. 41, pp. 401-409 (2000).

[71] Fraser H. J., and J. A. George, *Integrated container loading software for pulp and paper industry*, Eur. J. Oper. Res., vol. 77, pp. 466474 (1994).

[72] Garey M. R., and D. S. Johnson, Computers and intractability: a guide to the theory of NP-completeness, *San Francisco*, Freeman, (1979).

[73] Gensane T., *Dense packings of equal spheres in a larger sphere: a list of result*, in Les Cahiers du LMPA J. Liouville, vol. 188, (June 2003).

[74] Gensane T., *Dense Packings of Equal Spheres in a Cube*, Electronic J. Combinatorics, vol. 11(1), R33, (2004).

[75] George J. A., J. M. George, and B. W. Lamar, *Packing Different sized circles in to a rectangular container*, European Journal of Operation Research, vol. 84, pp. 693-712(1995).

[76] Gill P. E., W. Murray, and M. A. Saunders, *SNOPT*: An (SQP) Algorithm for Large-Scale Constrained Optimization, SIAM J. Optim., vol. 12, pp. 979-1006 (2002).

[77] Giunta A. A., S. F. Wojtkiewicz, and M. S. Eldred, *Overview of modern design of experiments methods for computational simulations*, AIAA 2003-0649 , pp. 1-17(2003).

[78] Glover F., *Heuristics for integer programming using surrogate constraints*, Decision Sciences, vol. 8, pp. 156-166 (1977).

[79] Glover F., *Future paths for integer programming and links to artificial intelligence*, Computers & Operations Research, vol. 13, pp. 533-549, (1986).

[80] Glover F., *Tabu search: Part I*, ORSA Journal on Computing, vol. 11(3), pp. 190206 (1989).

[81] Glover F., and M. Laguna, *Tabu search*, Modern heuristic techniques for combinatorial problems (Ed.: Reeves C.R.), Oxford: Blackwell, pp. 70150 (1993).

[82] Glover F., and M. Laguna, Tabu Search, *Kluwer Academic Publishers*, Boston, MA, (1997).

[83] Goldberg D. E., Genetic Algorithms in Search, *Optimization and Machine Learning, Addison- Wesley*, (1989).

[84] Goldberg M., *The packing of equal circles in a square*, Math Mag., vol. 43, pp. 24-30(1970).

[85] Goldberg M., *On the densest packing of equal spheres in a cube*, Math. Mag., vol. 44, pp. 198208 (1971).

[86] Graham R. L., *Sets of points with given minimum separation (Solution to Problem El921)*, Amer. Math. Monthly, vol. 75, pp. 192-193 (1968).

[87] Graham R. L., and B. D. Lubachevsky, *Dense packings of equal disks in an equilateral triangle: from 22 to 24 and beyond*, Electronic J. Combinatorics, vol. 2,(1995).

[102] Harter A., A. Hopper, P. Steggles, A. Ward, and P. Webster, *The anatomy of a context-aware application*, In Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 1999), Seattle, WA, ACM Press, pp. 59-68(August 1999).

[103] Hendrickson B. A., The molecular problem: Determining Conformation from pairwise distances, *PhD thesis*, Cornell University, (1991).

[104] Hertog D., and H. P. Stehouwer, *Optimizing color picture tubes by high-cost nonlinear programming*, European Journal on Operations Research , vol. 140(2), pp. 197-211(2002).

[105] Hifi M., and R. M'Hallah, *Approximate algorithms for constrained circular cutting problems*, Computers and Operations Research, vol. 31, pp. 675694(2004).

[106] Hifi M., and R. M'Hallah, *Adaptive and restarting techniques-based algorithms for circular packing problems*, Computational Optimization and Applications, vol. 39, pp. 17-35 (2008).

[107] Hochbaum D. S., and W. Maass, *Approximation schemes for covering and packing problems in image processing and VLSI*, Journal of the Association for Computing Machinery, vol. 1(32), pp. 130136 (1985).

[108] Holland J., *Adaptation In Natural and Artificial Systems*, University of Michigan Press, (1975).

[109] Howard A., M. J. Mataric, and G. S. Sukhatme, *Relaxation on a mesh: a formalism for generalized localization*, In Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), Wailea, Hawaii, (October 2001).

[110] Huang H., W. Huang, Q. Zhang, and D. Xu, *An improved algorithm for the packing of unequal circles within a larger containing circle*, European Journal of Operational Research, vol. 141, pp. 440-453 (2002).

[111] Huang W. Q., Z. P. Lu, and H. Shi, *Growth algorithm for finding low energy configurations of simple lattice proteins*, Physical Review E., vol. 72 (2), pp. 016704.1-016704.6 (2005).

[112] Huang W. Q., and Y. Kang, *A heuristic quasi-physical strategy for solving disks packing problem*, Simulation Modeling Practice and Theory, vol. 10, pp. 195207 (2002).

[113] Huang W., Y. Li, B. Jurkowiak , C. M. Li, and R. C. Xu, *A two-level search strategy for packing unequal circles into a circle container*, in Proceedings of the International Conference on Principles and Practice of Constraint Programming, Springer-Verlag, pp. 868-872 (2003).

[114] Huang W., Y. Li, C. M. Li, and R. C. Xu, *New heuristics for packing unequal circles into a circular container*, Computers and Operations Research, vol. 33, pp. 21252142 (2006).

[115] Hsu, H. P., V. Mehra, W. Nadler, and P. Grassberger, *Growth-based optimization algorithm for lattice heteropolymers*, Physical Review , vol. 68, pp. 021113(2003).

[88] Graham R. L., and B. D. Lubachevsky, *Repeated patterns of dense packings of equal disks in a square*, The Electronic J. of Comb., vol. 3, pp. 1-16 (1996).

[89] Graham R. L., B. D. Lubachevsky , K. J. Nurmela, and P. R. J. Osterggllrd, *Dense packings of congruent circles in a circle*, Discrete Mathematics, Elsevier, vol. 181, pp. 139-154(1998).

[90] Grassberger P., *Pruned-enriched Rosenbluth method: Simulations of θ polymers of chain length up to 1 000 000*,Phys. Rev., vol 56(3), 3682 - 3693 (1997).

[91] Groot de C., R. Peikert, and D. Wü rtz, *The optimal packing of ten equal circles in a square*, IPS Research Report 90-12, ETH Zü rich,(1990).

[92] Groot de C., R. Peikert, D. Würtz, and M. Monagan, *Packing circles in a square: a review and new results*, System Modelling and Optimization, Proc. 15th IFIP Conf. Zürich, pp. 45-54 (1991).

[93] Groot de C., D. Würtz, M. Hanf, K. H. Hoffmann, R. Peikert, and T. Koller, *Stochastic optimization - efficient algorithms to solve complex problems*, System Modeling and Optimization (Proc. 15th IFIP Conf. Z"'urich 1991), Lecture Notes in Control and Information Sciences(Ed.: P. Kall) Springer-Verlag, Berlin, vol. 180, (1992).

[94] Grosso A., M. Locatelli, and F. Schoen, *Solving Molecular Distance Geometry Problems By Global Optimization Algorithms*(2006). Available in www.optimization-online.org/DB_FILE/2006/09/1467.pdf,

[95] Grosso A. , M. Locatelli, and F. Schoen, *A Population Based Approach for Hard Global Optimization Problems Based on Dissimilarity Measures*, On-line submission, Optimization On line.(also published in Mathematical Programming: Series A and B archive, vol. 110(2), pp. 373 - 404 (2007).

[96] Grosso A., A. R. J. U. Jamali, and M. Locatelli, *Finding Maximin Latin Hypercube Designs by Iterated Local Search Heuristics*(2008). To appear in European Journal of Operations Research, Elsevier.

[97] Haessler R. W., and P. E. Sweeney, *Cutting stock problems and solution procedures*, European Journal of Operational Research, vol. 54(1), pp. 141150(1991).

[98] Hales T. C., *The Sphere Packing Problem*, J. Comput. Appl. Math., vol. 44, pp. 41-76 (1992).

[99] Ham,V. F., J. J. V. Wijk, *Beam trees: Compact visualization of large hierarchies*, In Proc. Information Visualization 2002, IEEE, pp. 3139 (2002).

[100] Hamming R. W., *Error Detecting and Error Correcting Codes*, Bell System Technical Journal, vol. 26(2), pp. 147-160 (1950).

[101] Harter A., and A. Hopper, *A distributed location system for the active office*, In IEEE Network, IEEE Computer Society Press, pp. 62-70 (January/February 1994).

[116] Husslage B., E. R. van Dam, and D. den Hertog, *Nested maximin latin hypercube designs in two dimensions*, CentER Discussion Paper No. 200579 (2005).

[117] Husslage B., G. Rennen, E. R. van Dam, and D. den Hertog, *Space-Filling Latin Hypercube Designs for Computer Experiments*, CentER Discussion Paper No. 2006-18 (2006).

[118] Hwan I. Y., *Uncertainty and sensitivity analysis of time-dependent effects in concrete structures*, Engineering Structures, vol. 29, pp. 13661374 (2007).

[119] Iman R. L., and Conover W. J., *A distribution-free approach to inducing rank correlation among input variables*, *Comm. Stat. Part B - Simulation Comput.*, vol. 11, pp. 311-334 (1982).

[120] Iman R. L., and J. C. Helton, *A comparison of uncertainty and sensitivity analysis techniques for computer models*, Report NUREG/CR-3904, SAND84- 1461. Albuerque: Sandia National Laboratories, (1985).

[121] Jacoby, J. E., and S. Harrison, *Multi-variable Experimentation and Simulation Models*, Naval Research Logistics Quarterly, vol. 9, pp. 121-136 (1962).

[122] Jin R., W. Chen, and A. Sudjianto, *An efficient algorithm for constructing optimal design of computer experiments*, Journal of Statistical Planning and Inference, vol. 134(1), pp. 268-287(2005).

[123] Johnson D. S., *Local optimization and the travelling salesman problem*, In Proceedings of the 17th Colloquium on Automata, Languages, and Programming, LNCS, Springer Verlag, Berlin, vol. 443, pp. 446461 (1990).

[124] Johnson D. S., and L. A. McGeoch, *The travelling salesman problem: A case study in local optimization*, Local Search in Combinatorial Optimization(Eds: Aarts E. H. L. and J. K. Lenstra), John Wiley & Sons, Chichester, England, pp. 215310 (1997).

[125] Johnson M. E., L. M. Moore, and D. Ylvisaker, *Minimax and maximin distance designs*, Journal of Statistical Planning and Inference, vol. 26, pp. 131-148(1990).

[126] Jones D., Schonlau, M., and W. Welch, *Eficient global optimization of expensive black-box functions*, Journal of Global Optimization, vol. 13, pp. 455-492 (1998).

[127] Joseph V. R., and Y. Hung, *Orthogonal-Maximin Latin Hypercube Designs*, Statistica Sinica, vol. 18, pp. 171-186 (2008).

[128] Jurecka F. , K.-U. Bletzinger, and M. Ganser, *Update Schemes for Global Approximations in Robust Design Optimization*, 6th World Congresses of Structural and Multidisciplinary Optimization, Rio de Janeiro, Brazil,(2005).

[129] Kiefer J., *Construction and optimality of generalized Youden designs*, A Survey of Statistical Design and Linear Models (Ed: Srivastava), North-Holland, Amsterdam, pp. 333-353 (1975).

[130] Kirchner K., and G. Wengerodt, *Die dichteste Packung von 36 Kreisen in einem Quadrat*, Beiträge Geom., vol. 25, pp. 147-159(1987).

[131] Kirkpatrick S., C. D. Gelatt Jr., and M. P. Vecchi, *Optimization by Simulated Annealing*, Science, vol. 220, pp. 671-680 (1983).

[132] Kleijnen, J. P. C., *Sensitivility analysis and related analysis: a review of some statistical techniques.*, J. Statist. Comp. Simul., vol. 57, pp. 111-42(1997).

[133] Koehler J. R., and A. B. Owen, *Computer experiments*, Handbook of Statistics, Elsevier Science B.V, vol. 13, (1996).

[134] Kojima M., S. Kim, and H. Waki, *Sparsity in Sums of Squares Polynomials*, Mathematical Programming, vol. 103(1), pp. 45-62(2006).

[135] Kottwitz D. A., *The densest packing of equal circles on a sphere*, Acta Cryst. Sect. A., vol. 47, pp. 158-165(1991).

[136] Kravitz S., *Packing cylinders into cylindrical containers*, Math. Mag., vol. 40, pp. 65-71 (1967).

[137] Krige D. G, A statistical approach to some mine valuations and allied problems at the Witwatersrand, *Master's thesis* ,University of Witwatersrand, (1951).

[138] Krumm J., S. Harris, B. Meyers, B. Brumitt, M, Hale, and S. Shafer, *Multi-camera multi-person tracking for easy living*, IEEE Workshop on Visual Surveillance, IEEE Press, pp. 3-10 (July 2000).

[139] Kubach T., A. Bortfeldt, and H. Gehring, *Parallel greedy algorithms for packing unequal circles into a strip or rectangle*(1996). Available in www.fernuni-hagen.de/wiwi/forschung/beitraege/pdf/db396.pdf

[140] Laurent M., *Matrix completion problems*, The Encyclopedia of Optimization, vol. 3, pp. 221-229(2001).

[141] Leary R. H., *Global optimization on funneling landscapes*, J. Global Optim., vol. 18, pp. 367383 (2000).

[142] Leary S., A. Bhaskar, and A. Keane, *Optimal orthogonal-array-based Latin hypercubes*, Journal of Applied Statistics, vol. 30(5), pp. 585598(2003).

[143] Lee C. Y. , *Some properties of non binary error-correcting codes*, IRE Transactions on Information Theory, vol. 4, pp. 77-82(1958).

[144] Lee J., H. A. Scheraga, and S. Rackovsky, *New optimization method for conformational energy calculations on polypeptides: conformational space annealing*, J Comput. Chem., vol. 18(9), pp. 1222-1232 (1997).

[145] Lee T. H., and J. J. Jung, *Maximin Eigenvalue Sampling of Kriging model*, 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 30 August - 1 September 2004, Albany, New York, (2000).

[146] Lee J., I. H. Lee, and J. Lee, *Unbiased global optimization of Lennard-Jones clusters for N: N ≤201 using the conformational space annealing method*, Physical Review Letters, vol. 91(8), pp. 080201/1-4 (2003).

[147] Lenstra J. K.,and A.H.G. R. Kan, *Complexity of packing, covering, and partitioning problems*, Packing and Covering in Combinatorics (Ed: Schrijver A.), Mathematisch Centrum, Amsterdam, pp. 275291(1979).

[148] Leung T. W., C. K. Chan , and M. D. Troutt, *Application of a mixed simulated annealing-genetic algorithm heuristic for the two-dimensional orthogonal packing problem.* European Journal of Operational Research, vol. 145, pp. 530-542 (2003).

[149] Li W., and Q. Y. Kenny, *Optimal Symmetric Latin Hypercube Designs.* Available in http://www.csom.umn.edu/WWWPages/Faculty/WLi/research/super.ps.

[150] Li W. W., and C. F. J. Wu, *Columnwise-Pairwise Algorithms With Applications to the Construction of Supersaturated Designs*, Technometrics, American Society for Quality Control and American Statistical Association, Alexandria, Va, USA, vol. 39(2), pp. 171-179(1997).

[151] Liefvendahl M., and R. Stocki, *A study on algorithms for optimization of Latin hypercubes*, Journal of Statistical Planning and Inference, vol. 136 (9), pp. 3231-3247(1 September 2006).

[152] Liu Y., and D. Chin, *A Novel Greedy Computing Algorithm for Rectangle Packing Problems*,International Journal of Computer Science and Network Security, vol. 6(4), pp. 78-81(2006).

[153] Lin D. K. J., and D. M. Steinberg, *A Construction Method for Orthogonal Latin Hypercube Designs*, Biometrika, Oxford University Press, vol. 93(2), pp. 279 -288(2006).

[154] Lin S., and B. W. Kernighan, *An effective heuristic algorithm for the traveling salesman problem.* Operations Research, vol. 21, pp. 498-516(1973).

[155] Locatelli M., and U. Raber, *Packing equal circles in a square: A deterministic global optimization approach*, Discrete Applied Mathematics, vol. 122(1-3), pp. 139-166(2002).

[156] Lodi A., S. Martello, and M. Monaci, *Two-dimensional packing problems: a survey*, European Journal of Operational Research , vol. 141, pp. 24152(2002).

[157] Lourenco H. R., O. Marting, and T. Stutzl, *A beginner's introduction to Iterated Local Search*, In Proceedings of MIC'2001-Meta-heuristics International Conference, Porto-Portugal, vol. 1, pp. 1-6(2001).

[158] Lourenco H. R. , O. C. Martin, and T. Stutzle, *In Iterated Local Search Handbook of Metaheuristics*, ISORMS 57(Eds.: Glover F., and G. Kochenberger,), Kluwer, pp. 321-353 (2002).

[159] L ü Z., and W. Huanga, *PERM for solving circle packing problem*, Computers & Operations Research, vol. 35(5), pp. 1742-1755 (2008).

[160] Lubachevsky D., *How to simulate billiards and similar systems*, J. Computational Physics, pp. 255-283 (1991).

[161] Lubachevsky B. D., and R. L. Graham, *Curved hexagonal packing of equal disks in a circle*, Discrete & Computational Geometry, pp. 179194 (1997).

[162] Lubachevsky B. D. , R. L. Graham, and F. H. Stillinger, Spontaneous Patterns in Disk Packings in Bridges: Mathematical Connections in Art, Music, and Science,(Ed. : Sarhangi R.), Gilliland Printing, Arkansas City, Kansas, pp. 181-193 (1998).

[163] Lubachevsky D., and F. H. Stillinger, *Geometric properties of random disk packing*, J. Statistical Physics, vol. 60, pp. 561-583 (1990).

[164] Markót M. Cs., *Optimal Packing of 28 Equal Circles in a Unit Square the First Reliable Solution*, Numerical Algorithms, vol. 37, pp. 253261(2004).

[165] Markót, M. Cs., and T. Csendes, *A new verified optimization technique for the "packing circles in a unit square" problems*, SIAM Journal on Optimization, vol. 16(1), pp. 193-219(2005).

[166] Marques, V. M. M., C. F. G. Bispo, and J. J. S. Sentieiro, *A system for the compaction of two dimensional irregular shapes based on simulated annealing*, IECON'91(IEEE), pp. 1911-1916(1991).

[167] Martin O., S. W. Otto, and E. W. Felten, *Large-step Markov chains for the traveling salesman problem*, Complex Systems, vol. 5(3), pp. 299326(1991).

[168] Martin O., and S. W. Otto, *Combining simulated annealing with local search heuristics*, Annals of Operations Research, vol. 63, pp. 5775, (1996).

[169] McKay M. D., Beckman, R. J., and W. J. Conover, *A comparison of three methods for selecting values of input variables in the analysis of output from a computer code*, Technometrics, vol. 21, pp. 239-245(1979).

[170] Melissen H, Packing and covering with circles, *PhD thesis*, Utrecht University ,(1997).

[171] Melissen H., *Loosest circles coverings of an equilateral triangle*, Math. Max., vol. 70, pp. 65-81 (1997).

[172] Melissen H., and P. C. Schuur, *Improved covering of a square with six and eight equal circles*, Electron J. Combin., vol. 3, (1996).

[173] Melissen H., Packing and covering with circles, *PhD thesis*, Univer. Utecht, (1997).

[174] Mladenovic N., F. Plastria, and D. Urosevic, *Reformulation descent applied to circle packing problems*, Computers & Operations Research, vol. 32(9), pp. 2419-2434(2005).

[175] Mollard M., and C. Payan, *Some progress in the packing of equal circles in a square*, Discrete & Computational Geometry, vol. 18, pp. 111-120, (1997).

[176] Montgomery D. C., Design and analysis of experiments, *John Wiley & Sons (Second edition)*, New York, (1984).

[177] Morris M. D., *Factorial plans for preliminary computational experiments*, Technometrics, vol. 33, pp. 161-74 (1991).

[178] Morris M. D., and T. J. Mitchell, *Exploratory designs for computer experiments*, Journal of Statistical Planning and Inference, vol. 43, pp. 381-402(1995).

[179] Moré J., and Z. Wu, *ϵ-optimal solutions to distance geometry problems via global continuation*, Preprint MCSP5200595, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, I ll, (May 1994).

[180] Moré J., and Z. Wu, *Global continuation for distance geometry problems*, SIAM Journal on Optimization, vol. 7, pp. 814836, (1997).

[181] Morrice, D. J., *A comparison of frequency domain methodology and conventional variable screening methods*, Operations Research Letters, vol. 17, pp. 165-174, 1995.

[182] Moser L., *Problem 24 (corrected)*, Canadian Mathematical Bulletin, vol. 3,(1960).

[183] Myers R. H., *Response surface methodology – Current status and future directions*, Journal of Quality Technology, vol. 31, pp. 30-74(1999).

[184] Nurmela K. J., *Constructing spherical codes by global optimization methods*, Research Report, A 32, Digital Systems Laboratory, Helsinki University of Technology, (1995).

[185] Nurmela K. J., *Conjecturally optimal coverings of an equilateral triangle with up to 36 equal circles*, Experiment. Math., vol. 9(2), pp. 241-250(2000).

[186] Nurmela K. J., and P. R. J. Östergård, *Packing up to 50 Equal Circles in a Square*, Discrete & Computational Geometry, vol. 18, pp. 111120(1997).

[187] Nurmela K. J., and P. R. J. Östergård, *More optimal packings of equal circles in a square*, Discrete and Computational Geometry, vol. 22, pp. 439-547(1999).

[188] Nurmela K. J., and P. R. J. Östergård, *Covering A Square with up to 30 equal circles*,(2000). Available in http://www.tcs.hut.fi/pub/reports/A62.ps.gz.

[189] Oliveira J. F. S., and J. A. S Ferreira, *Algorithms for nesting problems*, Applied Simulated Annealing, (Ed.: Vidal R. V.), Sringer Verlag, pp. 255-275(1992).

[190] Olsson A., Sandberg G., and O. Dahlblom, *On Latin hypercube sampling for structural reliability analysis* , Structural Safety, Elsevier, vol. 25(1), pp. 47-68(January 2003).

[191] Orr R. J., and G. D. Abowd, *The smart floor: A mechanism for natural user identiffcation and tracking*, Proceedings of the 2000 Conference on Human Factors in Computing Systems (CHI 2000), The Hague, Netherlands, ACM, (April 2000).

[192] Owen A., *Controlling correlations in Latin hypercube samples*, Journal of the American Statistical Association, vol. 89, pp. 1517-1522 (1994).

[193] Park J. S., *Optimal latin hypercube designs for computer experiments*, Journal of Statistical Planning and Inference, vol. 39, pp. 95-111(1994).

[194] Peikert, R., D. Würtz,, M., Monagan, and C. de Groot, *Packing Circles in a Square: A Review and New Results*, System Modeling and Optimization, Lecture Notes in Control and Information Sciences (Ed.: Kall P.), Springer-Verlag, Berlin, vol. 180, pp. 45-54 (1992).

[195] Peikert R., D. Würtz, M. Monagan, and C. den Groot, *Packing circles in a sphere: A review and new results*, Proceedings of the 15th IFIP Conference on System Modeling and Optimization, Lecture Notes in Control and Information Sciences, Springer, vol. 180, pp. 111-124(1991).

[196] Pintér J. D., and F. J. Kampas, *Nonlinear optimization in Mathematica with Math. Optimizer Professional*, Mathematica in Education and Research, vol. 10, pp. 1-18 (2005).

[197] Pisinger D., *Heuristics for the container loading problem*, European Journal of Operational Research, vol. 141, pp. 382392 (2002).

[198] Ponomarenko L. D., and P. M. Makmak, *New approaches to minimization on permutations when packing geometric objects*, Theory and Methods of Automated Design, Inst. for Technical Cybernetics, Ac. Sci. BSSR, Minsk, vol. 4, pp. 814 (1980).(in Russian)

[199] Priyantha N. B., Anit Chakraborty, and Hari Balakrishnan, *The cricket location-support system*, In Proceedings of MOBICOM 2000, pages 32-43, Boston, MA, ACM, ACM Press, (August 2000.).

[200] Reis G. E., *Dense packing of equal circles within a circle*, Math. Mag., vol. 48, pp. 33-37(1975).

[201] Rikards R., and J. Auzins, *Response surface method for solution of structural identification problems*, Inverse Problems in Engineering, vol. 12(1), pp. 59-70(2004).

[202] Rossi-Doria O., M. Samples, M. Birattari, M. Chiarandini, J. Knowles, M. Manfrin, M. Mastrolilli, L. Paquete, B. Paechter, and T. Stutzle, *A Comparison of the performance of different metaheuristics on the timetabling problem*, In Proceedings of PATAT 2002, The 4th international conference on the Practice and Theory of Automated Timetabling, Gent, Belgium, pp. 115-119(2002).

[203] Sacks J., W. J. Welch, T. J. Mitchell, and H. P. Wynn, *Design and analysis of computer experiments*, Statist. Sci., vol. 4, pp. 409-423(1989).

[204] Sacks J., S. B. Schiller, and W. J. Welch, *Designs for computer experiments*, Technometrics, vol. 31, pp. 41-47(1989).

[205] Santner T. J., B. J. Williams, and W. I. Notz, *The design and analysis of computer experiments*, Springer Series in Statistics, Springer-Verlag, New York ,(2003).

[206] Satterthwaite F. E., *Random Balance Experimentation*, Technometrics, vol. 1(2), pp. 111-137, (May 1959).

[207] Saxe J. B., *Embeddability of graphs in k-space is strongly NP-hard*, in 17th Allerton Conference in Communication, Control and Computing, pp. 480-489 (1979).

[208] Schaer J., *The densest packing of nine circles in a square*, Canadian Mathematical Bulletin, vol. 8, pp. 273277(1965).

[209] Schruben L. W., *Simulation Optimization Using Frequency Domain Methods*, Proceedings of the 1986 Winter Simulation Conference, pp. 369-396 (1986).

[210] Schürmann A. , *On Parametric Density of Finite Circle Packings*, (1999). Available in www.math.unisiegen.de/wills/...achill2.ps

[211] Schaer J., A. Meir, *On a geometric extremum problem*, Canadian Mathematical Bulletin, vol. 8, pp. 2127(1965).

[212] Sebastiani P., and H. P. Wynn, *Maximum entropy sampling and optimal Bayesian experimental design*, J. R. Statist. Soc., vol. 62(1), pp. 145-157 (2000).

[213] Sharma R., T. Balachander, C. McCord, S. Anand, and Q. Zhang, *Genetic Algorithms for the Single-Sheet and Multi-Sheet Non-convex Cutting Stock Problem*, Annual report, University of Cincinnati, (1997).

[214] Shewry M., and H. Wynn, *Maximum entropy design*, Journal of Applied Statistics, vol. 14, pp. 165-170 (1987).

[215] Simpson T. W., D. K. J. Lin, and W. Chen, *Sampling strategies for computer experiments: Design and analysis*, International Journal of Reliability and Applications, vol. 2(3), pp. 209-240(2001).

[216] Simpson T. W., A. J. Booker, D. Ghosh, A. A. Giunta, P. N. Koch, and R. -J. Yang, *Approximation methods in multidisciplinary analysis and optimization: A panel discussion*, Structural and Multidisciplinary Optimization, vol. 27(5), pp. 302-313(2004).

[217] Sobieski -S. J., and R. T. Haftka, *Multidisciplinary aerospace design optimization: Survey of recent developments*, Structural and Multidisciplinary Optimization, vol. 14(1), pp. 1-23(1997).

[218] Stein M., *Large sample properties of simulation using Latin hypercube sampling*, Technometrics, vol. 29, pp. 143-51 (1987).

[219] Steinberg G. D. M, and K. J. N. Dennis, *A construction method for orthogonal Latin hypercube designs*, Biometrika, vol. 93 (2), pp. 279288 (2006).

[220] Srivastava, J. N., *Design for Searching Non-Negligible Effects*, A Survey of Statistical Design and Linear Models, (Ed: Srivastava J. N.), North-Holland, Amsterdam, pp. 507-519 (1975).

[221] Stinstra E. D., D. den Hertog, H. P. Stehouwer, and A. Vestjens, *Constrained maximin designs for computer experiments*, Technometrics, vol. 45(4), pp. 340-346(2003).

[222] Stocki R., *A method to improve design reliability using optimal Latin hypercube sampling.* CAMES, vol. 12, pp. 393-411 (2005).

[223] Stoyan Y., and G. Yaskow, *Mathematical model and solution method of optimization problem of placement of rectangles and circles taking into account special constraints*, International Transactions in Operational Research, vol. 5, pp. 45-57 (1998).

[224] Stoyan, Y. G., *Mathematical methods for geometric design*, In: Advances in CAD/CAM: Proceedings of PROLAMAT82, Leningrad, USSR, North-Holland, Amsterdam, pp. 6786(2003).

[225] Stoyan Y. G., 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, vol. 156, pp. 590-600 (2004).

[226] Stützle T., Local Search Algorithms for Combinatorial Problems - Analysis, Improvements, and New Applications, *PhD thesis*, Darmstadt University of Technology, Department of Computer Science, (1998).

[227] Stützle T., *Applying Iterated Local Search to the permutation flow shop problem*, Technical Report AIDA-98-04, FG Intellektik, TU Darmstadt, (1998).

[228] Stützle T., and H. H. Hoos, *Analyzing the run-time behaviour of iterated local search for the TSP*, Technical Report IRIDIA/2000-01, IRIDIA, Université Libre de Bruxelles, (2000). Available at http://www.intellektik.informatik.tu-darmstadt.de/~tom/pub.html.

[229] Sugihara, K., M. Sawai, H. Sano, D. S. Kim, and D. Kim, *Disk packing for the estimation of the size of wire bundle.* Jpn. J. Ind. Appl. Math., vol. 21, pp. 259278 (2004).

[230] Sweeney, P. E., and E. R. Paternoster, *Cutting and packing problems: a categorized applications-oriented research bibliography.* J. Oper. Res. Soc., vol. 43, pp. 691706 (1992).

[231] Szabó P. G., *Some new structures for the equal circles packing in a square problem*, Central European Journal of Operations Research, vol. 8, pp. 79-91 (2000).

[232] Szabó, P. G., T. Csendes, L. G. Casado, and I. Garcá, *Packing Equal Circles in a Square I. Problem Setting and Bounds for Optimal Solutions*, Optimization Theory: Recent Developments from Mátraháza (Eds.: Giannessi F., P. Pardalos, and T. Rapcsák), Kluwer Academic Publishers, Dordrecht, Boston, London, pp. 191-206 (2001).

[233] Szabó P. G., M. C. Markót, and T. Csendes, *Global optimization in geometry - circle packing into the square*, Essays and Surveys in Global Optimization (Eds.: Audet C., P. Hansen, and G. Savard), Kluwer, pp. 233-266 (2005).

[234] Szabó P. G., Markót M. C., T. Csendes, E. Specht, L. G. Casado, and I. Garcia, *New Approaches to Circle Packing in a Square*, Optimization and Its Applications, Springer, (2007).

[235] Szabó P. G., and E. Specht, *Packing up to 200 Equal Circles in a Square*, Models and Algorithms for Global Optimization (Eds.: Törn A. , and J. Źilinskas), Springer Optimization and Its Applications, vol. 4, pp. 141156(2007).

[236] Taguchi, G., *Introduction to Quality Engineering: Designing Quality into Products and Processes*, Asian Productivity Organization, Japan, (1988).

[237] Tang B. X., *A theorem for selecting oa-based latin hypercubes using a distance criterion*, Communications in Statistics–Theory and Methods, vol. 23, pp. 2047-2058(1994).

[238] Tang B. X., *Selection Latin hypercube designs using correlation criteria*, Statist. Sinica, vol. 8, pp. 965-77(1998).

[239] Tarnai T., and Z. Gasper, *Covering a square by equal circles*, Elem. Math., vol. 50, pp. 167-170 (1995).

[240] Törn, A., A. Zilinskas, *Global Optimization*, Lecture Notes in Computer Science, Springer Verlag, Heidelberg, vol. 350, (1989).

[241] Tóth F. L., *Punktverteilungen in einem Quadrat*, Studia Sci. Math., Hung., vol. 6, pp. 439-442(1971).

[242] Trosset M. W., *Applications of multidimensional scaling to molecular conformation*, Computing Science and Statistics, vol. 29, pp. 148152(1998).

[243] Trosset M. W., *Approximate maximin distance designs*, Proceedings of the Section on Physical and Engineering Sciences, Alexandria, VA, pp. 223-227(1999).

[244] Trosset M. W., *Distance matrix completion by numerical optimization*, Comput. Optim. Appl., vol. 17(1), pp. 1122 (2000).

[245] Trosset M. W., *Extensions of classical multidimensional scaling via variable reduction*, Computational Statistics, vol. 17(2), pp. 147162 (2002).

[246] Tseng P., *SOCP relaxation for nonconvex optimization*, presented at ICCOPT 1, RPI, Troy, NY, August 25, (2004).

[247] Wales D. J., and J. P. K. Doye, *Global optimization by basin-hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms*, J. Phys. Chem. A., vol. 101, pp. 51115116(1997).

[248] Wang G. G., *Adaptive Response Surface Method Using Inherited Latin Hypercube Design Points*, Transactions of the ASME, Journal of Mechanical Design, vol. 125, pp. 210-220(June 2003).

[249] Wang H., W. Huang , Q. Zhang, and D. Xu, *An improved algorithm for the packing of unequal circles within a larger containing circle*, European Journal of Operational Research, vol. 141, pp. 440-453 (2002).

[250] Wang W., H. Wang, G. Dai, and H. Wang, *Visualization of large hierarchical data by circle packing*, ACM Special Interest Group on Computer-Human Interaction, ACM New York, NY, USA, pp. 517 - 520 (2006).

[251] Want R., A. Hopper, V. Falcao, and J. Gibbons, *The active badge location system*, ACM Transactions on Information Systems, vol. 10(1), pp. 91-102, (January 1992).

[252] Want R., B. Schilit, N. Adams, R. Gold, K. Petersen, D. Goldberg, J. Ellis, and M. Weiser, *The PARCTAB ubiquitous computing experiment*, In Tomasz Imielinski, editor, Mobile Computing, Kluwer Publishing, February, ch. 2, pp. 45-101(1997).

[253] Wäscher G., H. Hauβner, H. Schumann, and Management, *An Improved Typology of Cutting and Packing Problems*, Working Paper No. 24, Faculty of Economics, Otto von Guericke Univ., Magdeburg, (2005).

[254] Weinberger K. Q., F. Sha, and L. K. Saul, *Learning a kernel matrix for nonlinear dimensionality reduction*, In ICML '04: Proceedings of the twenty-first international conference on Machine learning, ACM Press, 106(2004).

[255] Wengerodt G., *Die dichteste Packung von 16 Kreisen in einem Quadrat*, Beitrage zur Algebra und Geometrie, vol. 16, pp. 173-190 (1983).

[256] Wengerodt G., *Die dichteste Packung von 14 Kreisen in einem Quadrat*, Beitrage zur Algebra und Geometrie, vol. 25, pp. 25-46 (1987).

[257] Wengerodt G., *Die dichteste Packung von 25 Kreisen in einem Quadrat*, Ann. Univ. Sci. Budapest Eotvos Sect. Math., vol. 30, pp. 3-15 (1987).

[258] Wenqi H., X. Ruchu, *Two personification strategies for solving circles packing problem*, Science in China (Series E), vol. 29(4), pp. 34753 (1999).

[259] Wenqi H., Z. Shuhao, *A quasi-physical method for solving packing problems*, Acta Mathematicae Applagatae Sinica, vol. 2(2), pp. 176180 (1979).

[260] Wenqi H., Z. Shuhao, *A heuristic quasi-physical strategy for solving disks packing problem*, Simulation Modelling Practice & Theory, vol. 10(3), pp. 195-207(2002).

[261] Williams G. A., J. M. Dugan, and R. B. Altman, *Constrained global optimization for estimating molecular structure from atomic distances*, Journal of Computational Biology, vol. 8(5), pp. 523547(2001).

[262] Wu D., and Z. Wu, *An updated geometric build-up algorithm for molecular distance geometry problems with sparse distance data*, Journal of Global Optimization archive, vol. 37(4), pp. 661 - 673 (April 2007).

[263] Xu H. , *Universally optimal designs for computer experiments*, Statistica Sinica, vol. 9, pp. 1083-1088(1999).

[264] Xu Y., *On the minimum distance determined by n($\leq$ 7) points in an isosceles right triangle*, Acta Math. Appl. Sinica (English Ser.), vol. 12, pp. 169-175(1996).

[265] Ye K. Q., *Column orthogonal Latin hypercubes and their application in computer experiments*, Journal of the Americal Statistical Association, vol. 93, pp. 1430-1439(1998).

[266] Ye K. Q., W. Li, and A. Sudjainto, *Algorithmic construction of optimal symmetric Latin hypercube designs*, Journal of Statistical Planning and Inference, vol. 90, pp. 145-159(2000).

[267] Zahn C. T. Jr., *Black box maximization of circular coverage*, J. Res. Nat. Bur. Standards Sect, vol. 66, pp. 181-216 (1962).

[268] Zhang D., and W. Huang, *A simulated Annealing Algorithm for Circles Packing Problem*, ICCS 2004 (Eds.: Bubak M., et. al), Springer-Verlag, pp. 206-214 (2004).

[269] Zhang D., and A. Deng, *An effective hybrid algorithm for the problem of packing circles into a larger containing circle*, Computers & Operations Research, vol. 32, pp. 1941-1951 (2005).

[270] Zhou Y. M. , Arrangements of Points on the Sphere, *PhD thesis*, University of South Florida, (1995).

[271] http://www.fewcal.kub.nl/sturm/software/sedumi.html,

[272] http://www.home.att.net/donovanhse/Packing/index.html.

[273] http://www.hydra.nat.uni-magdeburg.de/packing/cci/cci.html.

[274] http://www.packomania.com. (updated 20-Mar-2008).

[275] http://www.pinpointco.com/(2001).

[276] http://www.spacefillingdesigns.nl/. (updated Sept. 25, 2008).

APPENDIX

A. OVERALL IMPROVED VALUES

A.1 Overall improved radii in ICPCC

Here we display all the improved values (radii of the circular containers) for ICPCC. For $n = 2, \ldots, 100$ our proposed algorithms are able to reach all the best known values in the literature (available in [274]) and to obtain 21 improved configurations, whose radii are reported in Table A.1.

A.2 Overall improved radii in NICPCC

Here we display all the improved values for NICPCC. As we mentioned in Chapter 9, we consider the instances given in [106]. For these instances our algorithms were able to reach all the best known results and to improve some of them, whose radii are reported in Table A.2.

Tab. A.1: The overall improved radii for ICPCC problem

n	radii
66	9.096279426924
67	9.168971881784
70	9.345653194084
71	9.415796896871
73	9.540346152138
74	9.589232764339
75	9.672029631947
77	9.798911924507
78	9.857709899885
83	10.116857875102
86	10.298701053110
87	10.363208505078
88	10.432337692732
89	10.500491814574
92	10.684645847916
93	10.733352600260
94	10.778032160252
96	10.883202759722
97	10.938590110073
99	11.033141151446
100	11.082149724310

Tab. A.2: The overall improved radii for NICPCC problem

Test n.	n	`OurBestResults`
12	11	60.7099
13	14	113.5552
14	17	49.1873
16	15	38.8380
18	162	11.5119

B. SOME IMPROVED SOLUTIONS

B.1 Examples of improved LHDs

In Tables B.1-B.4 we report few examples with the coordinates of LHDs which improve the best known ones reported in [276]

Tab. B.1: Examples of improved Maximin LHDs for $k = 3, 4, 5$ (improved w.r.t the values available in [276])

Pts	Factors/Coordinates		
$(N,k) = (17,3)$; Mm=[54,18]			
1	0	4	13
2	1	3	4
3	2	10	8
4	3	12	16
5	4	9	1
6	5	16	3
7	6	1	9
8	7	15	10
9	8	6	14
10	9	8	7
11	10	2	2
12	11	11	0
13	12	13	15
14	13	0	11
15	14	14	6
16	15	7	12
17	16	5	5
$(N,k) = (33,3)$; Mm=[122,1]			
1	0	14	9
2	1	25	13
3	2	17	21
4	3	5	17
5	4	8	29
6	5	28	24
7	6	4	4
8	7	26	3
9	8	15	1
10	9	19	30
11	10	32	14
12	11	12	12
13	12	0	25
14	13	22	18
15	14	11	23
16	15	1	11
17	16	29	27
18	17	20	7
19	18	9	2
20	19	6	32
21	20	31	8
22	21	18	28
23	22	3	20
24	23	16	16
25	24	30	19
26	25	23	0
27	26	2	6
28	27	27	31
29	28	13	5
30	29	10	26
31	30	24	10
32	31	21	22
33	32	7	15

Pts	Factors/Coordinates			
$(N,k) = (14,4)$; Mm=[79,8]				
1	0	3	5	7
2	1	9	11	4
3	2	12	4	9
4	3	8	2	1
5	4	4	12	11
6	5	1	10	2
7	6	5	3	13
8	7	0	0	6
9	8	13	7	3
10	9	11	9	12
11	10	7	13	5
12	11	10	1	8
13	12	2	8	10
14	13	6	6	0
$(N,k) = (36,4)$; Mm=[298,2]				
1	0	13	10	13
2	1	23	21	22
3	2	30	27	6
4	3	8	30	27
5	4	9	12	31
6	5	27	4	23
7	6	12	28	9
8	7	28	6	5
9	8	32	31	32
10	9	0	19	17
11	10	1	2	14
12	11	10	8	0
13	12	26	15	35
14	13	34	18	15
15	14	25	34	16
16	15	22	20	1
17	16	15	0	28
18	17	17	29	33
19	18	16	17	19
20	19	3	22	3
21	20	5	32	20
22	21	2	11	29
23	22	20	1	11
24	23	31	5	25
25	24	14	35	4
26	25	33	25	26
27	26	4	7	10
28	27	35	9	7
29	28	29	26	8
30	29	21	16	34
31	30	19	33	21
32	31	18	13	2
33	32	6	24	30
34	33	11	3	24
35	34	7	23	12
36	35	24	14	18

Pts	Factors/Coordinates				
$(N,k) = (11,5)$; Mm=[82,6]					
1	0	7	10	5	0
2	1	8	2	6	4
3	2	5	9	2	8
4	3	0	6	8	2
5	4	1	1	1	7
6	5	3	4	9	10
7	6	6	5	0	1
8	7	10	7	10	5
9	8	9	3	3	9
10	9	4	0	7	3
11	10	2	8	4	6
$(N,k) = (39,5)$; Mm=[575,2]					
1	0	33	12	23	20
2	1	9	28	8	13
3	2	11	4	16	26
4	3	18	25	13	36
5	4	21	22	36	7
6	5	6	31	30	24
7	6	29	37	20	15
8	7	26	18	12	0
9	8	35	23	2	22
10	9	17	13	35	35
11	10	5	7	9	4
12	11	1	10	32	14
13	12	22	0	27	6
14	13	37	27	29	32
15	14	27	1	5	17
16	15	32	9	14	38
17	16	10	35	24	3
18	17	8	14	0	27
19	18	7	38	10	28
20	19	19	19	19	19
21	20	36	6	34	23
22	21	24	34	1	8
23	22	0	17	21	34
24	23	38	21	25	5
25	24	15	33	31	37
26	25	25	32	38	16
27	26	31	36	11	29
28	27	2	26	6	9
29	28	14	16	33	1
30	29	20	11	3	2
31	30	12	8	37	25
32	31	16	2	15	33
33	32	4	5	17	12
34	33	34	15	4	21
35	34	3	29	28	18
36	35	28	3	22	11
37	36	30	20	26	31
38	37	13	24	7	30
39	38	23	30	18	10

Tab. B.2: Examples of improved Maximin LHDs for $k = 6, 7$
(improved w.r.t the values available in [276])

Pts	Factors/ Coordinates					
	$(N, k) = (9, 6)$; Mm=[82,6]					
1	0	8	5	5	4	4
2	1	1	2	3	2	0
3	2	0	6	7	7	5
4	3	2	8	1	1	6
5	4	3	0	4	3	8
6	5	4	4	0	8	3
7	6	5	1	8	6	1
8	7	7	3	2	0	2
9	8	6	7	6	5	7
	$(N, k) = (41, 6)$; Mm=[938,1]					
1	0	14	14	24	38	14
2	1	38	24	30	11	22
3	2	7	39	22	17	16
4	3	28	6	17	25	37
5	4	17	17	18	6	0
6	5	6	10	26	5	29
7	6	15	27	39	26	35
8	7	30	34	13	36	27
9	8	37	11	10	27	8
10	9	2	23	9	30	33
11	10	21	33	14	7	38
12	11	25	2	38	16	10
13	12	27	32	33	28	2
14	13	5	1	7	20	13
15	14	31	4	12	1	21
16	15	12	28	4	32	4
17	16	32	37	8	10	9
18	17	0	19	35	21	7
19	18	36	15	34	37	24
20	19	18	31	37	0	15
21	20	4	25	5	2	18
22	21	8	3	31	31	31
23	22	22	9	2	40	25
24	23	40	21	3	18	32
25	24	29	12	36	8	36
26	25	9	36	27	39	20
27	26	34	40	32	19	30
28	27	13	8	6	12	39
29	28	19	7	23	35	1
30	29	39	18	28	9	5
31	30	3	29	29	13	34
32	31	16	38	1	23	28
33	32	24	13	0	14	6
34	33	10	5	25	4	12
35	34	35	30	15	34	11
36	35	11	35	20	15	3
37	36	23	22	21	33	40
38	37	33	0	19	22	23
39	38	1	16	11	29	19
40	39	20	20	40	24	17
41	40	26	26	16	3	26

Pts	Factors/Coordinates						
	$(N, k) = (7, 7)$; Mm=[62,7]						
1	0	5	3	1	0	2	4
2	1	4	6	6	4	5	3
3	2	0	0	5	2	3	1
4	3	2	2	0	5	6	5
5	4	6	1	3	6	1	2
6	5	1	4	4	3	0	6
7	6	3	5	2	1	4	0
	$(N, k) = (43, 7)$; Mm=[1301,1]						
1	0	17	24	9	22	27	2
2	1	24	20	15	3	34	34
3	2	11	2	11	18	7	28
4	3	20	21	38	25	16	40
5	4	7	26	30	2	12	16
6	5	29	1	20	32	36	26
7	6	10	6	36	33	17	8
8	7	14	34	18	34	0	21
9	8	36	5	26	6	15	11
10	9	39	22	4	27	14	35
11	10	26	23	42	13	40	9
12	11	38	29	31	41	23	14
13	12	19	41	16	31	38	31
14	13	37	42	29	9	18	25
15	14	0	10	22	14	42	18
16	15	2	32	0	11	21	27
17	16	30	30	3	7	3	10
18	17	4	14	7	42	24	23
19	18	33	7	8	37	10	5
20	19	3	39	33	28	25	6
21	20	42	25	10	19	41	12
22	21	27	27	39	20	1	3
23	22	35	0	28	30	4	33
24	23	25	17	21	0	5	39
25	24	8	33	35	5	32	38
26	25	13	19	41	38	39	29
27	26	40	13	34	15	33	37
28	27	1	11	13	17	9	4
29	28	22	3	2	8	28	20
30	29	18	31	17	1	29	1
31	30	6	4	23	21	22	42
32	31	28	36	27	36	13	41
33	32	23	38	6	35	19	7
34	33	21	8	25	29	35	0
35	34	15	9	40	4	20	17
36	35	16	18	1	26	2	32
37	36	32	35	5	10	26	36
38	37	9	40	24	12	6	22
39	38	34	12	12	40	30	30
40	39	12	15	32	39	8	19
41	40	31	37	37	23	31	15
42	41	41	16	19	16	11	13
43	42	5	28	14	24	37	24

Tab. B.3: Examples of improved Maximin LHDs for $k = 8, 9$
(improved w.r.t the values available in [276])

Pts	Factors/ Coordinates								Pts	Factors/Coordinates								
	$(N,k) = (7,8)$; Mm=[71,1]									$(N,k) = (7,9)$; Mm=[80,1]								
1	0	6	2	3	0	5	3	4	1	0	6	1	5	1	4	2	3	5
2	1	1	3	6	4	4	6	1	2	1	1	5	2	5	6	6	4	4
3	2	0	4	4	3	2	0	6	3	2	4	4	0	4	2	0	6	1
4	3	3	1	0	6	1	5	5	4	3	5	3	3	6	0	5	0	3
5	4	4	6	1	5	6	2	2	5	4	0	0	6	3	1	3	5	2
6	5	2	0	2	1	3	1	0	6	5	3	2	1	0	5	4	1	0
7	6	5	5	5	2	0	4	3	7	6	2	6	4	2	3	1	2	6
	$(N,k) = (43,8)$; Mm=[1635,1]									$(N,k) = (43,9)$; Mm=[1957,1]								
1	0	30	8	12	14	8	8	28	1	0	11	26	5	22	11	13	16	40
2	1	23	34	27	37	36	23	11	2	1	28	42	28	33	16	11	13	9
3	2	2	18	35	17	20	38	22	3	2	17	35	29	19	41	23	39	28
4	3	26	31	36	3	17	14	6	4	3	31	17	31	9	14	36	3	32
5	4	1	36	13	12	14	7	26	5	4	16	9	37	42	15	31	27	27
6	5	21	12	3	31	24	42	29	6	5	14	10	35	12	34	3	8	19
7	6	28	27	20	5	31	28	42	7	6	2	30	33	6	6	22	26	11
8	7	25	13	26	36	2	34	5	8	7	37	0	13	21	31	18	28	36
9	8	11	14	4	19	29	18	0	9	8	35	4	26	20	2	14	22	3
10	9	42	7	21	9	25	36	10	10	9	15	20	0	37	18	28	33	5
11	10	33	41	5	28	6	19	14	11	10	36	15	22	28	40	35	12	4
12	11	15	19	7	32	39	5	36	12	11	7	11	9	1	39	34	20	21
13	12	7	11	34	33	15	2	13	13	12	33	31	6	3	17	9	37	15
14	13	17	3	30	4	41	11	20	14	13	3	37	20	30	27	42	6	20
15	14	12	32	22	41	10	24	41	15	14	30	25	41	15	9	6	30	39
16	15	29	4	41	30	26	27	37	16	15	20	27	4	7	12	26	1	2
17	16	41	30	33	29	16	1	30	17	16	18	19	16	11	8	40	42	37
18	17	0	0	14	15	18	20	40	18	17	38	39	1	29	28	33	23	33
19	18	34	39	37	21	11	40	24	19	18	1	5	18	24	23	5	41	24
20	19	37	33	9	8	37	6	15	20	19	25	28	32	40	37	16	7	41
21	20	16	25	6	0	5	35	19	21	20	10	33	3	25	42	4	15	17
22	21	36	2	11	42	22	12	16	22	21	5	16	25	38	5	7	2	18
23	22	14	20	39	7	0	15	34	23	22	42	21	12	18	19	1	0	25
24	23	8	42	1	25	33	31	23	24	23	26	8	38	2	29	25	40	12
25	24	18	29	28	6	42	39	9	25	24	40	22	27	36	33	2	36	14
26	25	10	38	40	20	35	13	31	26	25	39	34	36	26	10	37	32	13
27	26	4	40	29	22	9	25	3	27	26	13	2	17	0	3	12	14	30
28	27	35	16	42	26	32	16	1	28	27	4	23	39	32	35	19	24	0
29	28	5	9	23	39	38	30	17	29	28	29	40	34	4	36	21	10	16
30	29	38	10	15	24	1	33	35	30	29	6	41	23	39	13	17	35	31
31	30	39	28	19	35	40	29	32	31	30	12	6	2	31	26	29	4	34
32	31	32	17	18	13	3	4	4	32	31	0	18	40	13	25	30	18	38
33	32	6	15	2	34	4	17	18	33	32	9	7	24	23	4	41	19	6
34	33	13	1	31	11	13	32	7	34	33	32	12	8	34	0	15	31	29
35	34	27	5	0	10	34	26	25	35	34	21	38	19	16	1	27	5	35
36	35	19	35	17	38	30	3	8	36	35	23	1	7	17	30	10	21	1
37	36	24	37	8	16	12	10	39	37	36	22	14	21	35	38	38	38	26
38	37	3	24	16	2	28	9	12	38	37	41	13	11	5	21	39	17	22
39	38	31	26	10	27	19	37	2	39	38	34	3	42	27	20	20	11	23
40	39	40	22	32	1	23	22	27	40	39	8	36	10	8	24	32	34	10
41	40	9	23	24	18	21	41	38	41	40	24	24	15	10	32	8	29	42
42	41	20	6	25	23	27	0	33	42	41	19	29	30	14	7	0	25	8
43	42	22	21	38	40	7	21	21	43	42	27	32	14	41	22	24	9	7

Tab. B.4: Examples of improved Maximin LHDs for $k = 10$
(improved w.r.t the values available in [276])

Pts	Factors/ Coordinates									
	$(N, k) = (7, 10)$; Mm=[90,1]									
1	0	6	4	2	3	4	0	5	4	5
2	1	1	5	5	5	0	3	2	0	4
3	2	3	2	0	4	1	4	0	6	1
4	3	0	0	6	2	5	1	3	5	2
5	4	2	3	1	1	6	5	1	1	6
6	5	5	1	4	6	3	6	6	3	3
7	6	4	6	3	0	2	2	4	2	0
	$(N, k) = (10, 10)$; Mm=[174,1]									
1	0	7	7	2	1	5	7	4	7	0
2	1	9	5	7	8	1	4	2	1	6
3	2	2	3	3	9	8	2	9	3	2
4	3	6	1	4	0	6	0	5	5	9
5	4	0	9	5	2	7	6	3	0	7
6	5	3	0	9	3	2	9	7	4	3
7	6	1	8	6	5	0	1	6	9	4
8	7	5	4	8	7	9	5	0	8	5
9	8	8	6	1	6	4	8	8	6	8
10	9	4	2	0	4	3	3	1	2	1
	$(N, k) = (23, 10)$; Mm=[750,1]									
1	0	5	18	14	15	18	10	2	5	9
2	1	2	6	18	7	0	15	11	15	16
3	2	12	21	10	11	5	0	18	20	11
4	3	20	15	4	0	4	9	3	7	15
5	4	13	4	0	21	7	11	12	10	3
6	5	22	14	19	16	12	21	17	9	14
7	6	7	2	2	4	16	19	15	3	19
8	7	10	8	12	3	20	16	14	21	1
9	8	18	1	16	14	14	3	4	19	17
10	9	1	9	13	18	21	7	22	14	20
11	10	19	12	8	8	19	1	20	1	8
12	11	8	17	1	13	11	18	6	22	21
13	12	14	0	20	2	9	13	7	0	5
14	13	4	10	22	17	6	4	19	8	0
15	14	16	16	17	12	1	14	0	17	2
16	15	6	22	5	9	8	20	16	4	4
17	16	11	13	11	19	2	6	9	2	22
18	17	0	7	3	5	10	2	5	11	7
19	18	9	20	21	1	15	8	10	13	18
20	19	21	11	6	10	22	17	1	6	12
21	20	3	3	15	20	13	22	8	12	10
22	21	17	5	7	6	3	12	21	16	13
23	22	15	19	9	22	17	5	13	18	6

B.2 Examples of solutions for ICPCC

In Figure B.1 best known solutions for ICPCC with $n = 61, \ldots, 72$ are displayed. Some of these solutions, namely those with $n = 66, 67, 70, 71$, belong to those improved by our methods.

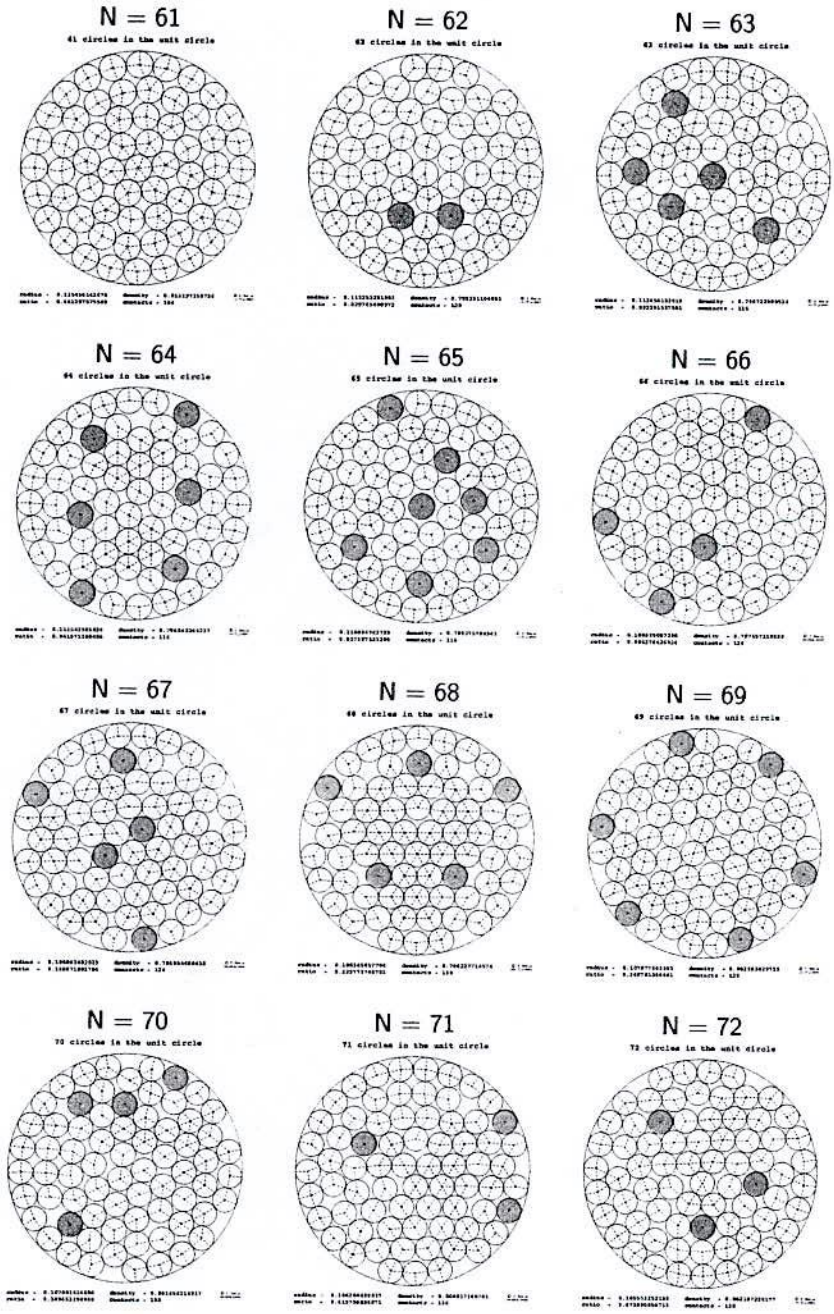

Fig. B.1: Few examples of best known solutions for ICPCC for $n = 61 - -72$, among which the solutions for $n = 66, 67, 70, 71$ have been obtained by our methods