Instance generator for the quadratic assignment problem with additively decomposable cost function

Madalina M. Drugan
Department of Information and Computing Sciences, Utrecht University,
PO Box 80.089, 3508 TB, Utrecht, The Netherlands
Artificial Intelligence Lab, Vrije Universiteit Brussel,
Pleinlaan 2, B-1050 Brussels, Belgium
Email: mdrugan@vub.ac.be

Abstract—Quadratic assignment problems (QAPs) are NP-hard combinatorial optimization problems often used to compare the performance of meta-heuristics. In this paper, we propose a QAP problem instance generator whose instances can be used as benchmark for meta-heuristics. Our generator aggregates various QAP instances into a larger size QAP instance in order to obtain an large size, additively decomposable QAP. We assume that a QAP instance which is difficult for local search has many local optima from which local search needs to escape from. We call the resulting QAPs the composite QAP instances (cQAPs). We use numerical and empirical techniques for the landscape analysis of generated composite QAPs. The comparison between our QAP instances with the other QAPs from the literature classify cQAPs as difficult. We show that heuristic algorithms that exploit the particularities of the cQAP search space, like iterated local search, can outperform other heuristics that do not consider the structure of this search space, like multi-restart local search.

I. INTRODUCTION

The Quadratic assignment problem (QAP) models many real-world problems like computer aided design in the electronics industry, scheduling, vehicle routing, etc. Recent extensive reviews on QAP instances are given in [11, 6]. Intuitively, QAPs can be described as the (optimal) assignment of a number of facilities to a number of locations. Let us consider N facilities, a set Π(N) of all permutations of {1, . . . , N} and the N × N distance matrix A = (aij), where aij is the distance between location i and location j. We assume a flow matrix B = (b ij) where b ij represents the flow from facility i to facility j. The goal is to minimize the cost function

\[ c(\pi) = \sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij} \cdot b_{\pi(i)\pi(j)} \]  

(1)

where \( \pi(\cdot) \) is a permutation. It takes quadratic time to evaluate this function. We consider a QAP as a tuple \((A, B, s)\) where s, if known, is the optimal solution. If the optimal solution is not known, we denote the QAP simply with \((A, B)\). Here, we consider QAP instances that were initially used in [13]. We pose three requirements for the flow and distance matrices. The distance and the flow between the same facilities is 0, \( \forall i, a_{ii} = 0 \) and \( b_{ii} = 0 \). The distance and the flow matrices are symmetrical, \( \forall i, j, a_{ij} = a_{ji} \) and \( b_{ij} = b_{ji} \). All the elements of the distance and flow matrices are non-negative, natural numbers within a range, \( \forall i, j, 0 \leq a_{ij} \in \mathbb{N} \) and \( 0 \leq b_{ij} \in \mathbb{N} \).

Related work. Drezner et al. [2] propose instances of QAPs that are difficult to solve with some heuristics, but easy to solve with exact methods. QAP instances proposed in [2] are simplified versions of instances from [10] where the underlying grid is very sparse and the distribution generating positions of facilities on this grid is random. The goal is to generate difficult QAPs with sparse flow matrices for heuristics. The QAP instance generator from [2] splits the facilities in regions, or clusters. Because of their sparseness, these QAP instances are easy to solve with some exact methods [3], even with about 90 facilities, and difficult for the heuristics that use the popular two facilities exchange operator [9].

Taillard [13] analysed the performance of some heuristics on several QAPs from the literature. He points out that, even though generating uniform randomly distributed values for distance and flow matrices is a popular choice, the resulting QAPs are not interesting for heuristics because their exact optimum is difficult to find, but the difference between some easy to find local optima and the global optimum value is small. Moreover, the difference in cost function between the best and the worst cost values is approaching 0 when the number of facilities approaches \( \infty \). This behaviour is explained by the lack of correlation between local optima. Taillard created a larger variation between the worst and the best local optimum using non-uniform random elements for distance and flow matrices. He concluded that large size QAPs with interesting structure to exploit by heuristic search algorithms are difficult to generate by a problem generator.

Local search algorithms. Intuitively, local search [12] starts from an initial solution and iteratively generates new solutions using a neighbourhood strategy. Each step, a solution that improves over the existing best-so-far solution is chosen. The algorithm stops when there is no improvement possible. Best improvement explores all the individuals in the neighbourhood of a solution and selects the best solution that is improving over the initial solution and all the other visited solutions. Because LS can be stuck in local optima, the multi-restart local search algorithm restarts LS multiple times from one uniform randomly chosen initial solution.

QAPs are permutation minimization problems. A suitable neighbourhood operator for QAPs is the 2-exchange swapping operator that swaps the position of two different facilities. This operator is attractive because of its linear time to compute the change in the cost function with the condition that the matrices A and B are symmetrical. The size of the neighbourhood increases quadratically with the number of facilities.
There are certain limitations in the design of multi-restart LS because it is basically random sampling in the space of local optima, it does not scale up for a large number of local optima. Iterated local search algorithms restart the search from solutions obtained by mutating local optimal solutions in order to restart LS from good regions of the search space.

In permutation problems like QAPs, the mutation operator interchanges facilities between different positions. When LS uses the 2-exchange operator to generate a neighbourhood, it interchanges facilities between different positions. When LS restarts LS from good regions of the search space.

The stopping criterion for the iterated LS is chosen to fairly compare its performance to the multi-restart LS. The search in the iterated LS is halted when it reaches the same number of swaps as the multi-restart LS. The distance between two solutions is defined as the minimum number of exchanges necessary to obtain one solution from another. The distance between a solution and its \( m \)-exchange solution is \( m - 1 \). Counting the number of swaps is equivalent to counting the number of function evaluations.

The main contributions of this paper. Our goal is to design a QAP instance generator that creates useful benchmark instances for heuristics like iterated LS which exploits the particularities of the search space. Our solution is to aggregate QAPs with computable optimal solutions, into a larger QAP and, afterwards, to fill the rest of the positions in the composite QAP with specific values. Note that there are two optimization problems corresponding with the two regions: i) the region of the component QAPs, or the composite region, and ii) the region outside these component QAPs, or the outside region. This class of QAP instances is denoted as composite QAPs.

We show that a composite QAP instance has an additively decomposable cost function that can be expressed as the sum of component QAP’s costs plus two extra terms corresponding to: i) the overlapping parts of the component QAPs, and ii) the outside region to the components QAPs. Problems with additively decomposable cost functions are considered useful test benchmark for meta-heuristic algorithm [11].

Approximating the number and the size of basins of attraction is a good method for landscape analysis [5]. The larger the number and the size of local optima, the more probable it becomes that local search gets stuck before finding the global optimum. We show that even small size (composite) QAPs have a lot of local optima and their number increases with the number of facilities. Due to the possible huge number of local optima, we propose an alternative definition and an empirical approximation of the distribution of basins of attraction.

We use iterated LS that restarts local search from correlated solutions, as opposed to multi-restart LS that restarts local search from random solutions, to connect the dynamical exploitation - the number of enhancements - and exploration - the probability of escaping local optimum - of the landscape with the search efficiency of iterated LS [4]. We show that iterated LS performs better on cQAPs than multi-restart local search because it exploits the structure of the search space.

Outline. Section [II] introduces an algorithm that generates composite QAP instances. Section [III] describes landscape analysis on several QAP instances. Section [IV] shows that iterated local search can outperform multi-restart local search on cQAPs. Section [V] concludes this paper.

II. A COMPOSITE QAP INSTANCES GENERATOR

In this section, we design an algorithm that generates QAPs with desired properties: i) large size, ii) many local optima, and iii) structure of the search space that can be exploited with genetic like operators. In Algorithm [II] the pseudo-code for generating composite QAPs, there are two function calls. The input is a set of \( d \) small size QAP matrices with identity permutation \( I \) as optimal solution into a larger size QAP, where \( \forall i \in \{1, \ldots, N\}, I(i) = i \). In general, the exact solution for a QAP is different from the identity permutation. A straightforward method to transform a component QAP with an optimal solution \( s \) into a QAP with identity permutation as optimal solution is to rename the facilities.

The function \( \text{assemble} \) aggregates the input component QAPs. In order to keep track of the assigned positions in the matrices of the composite QAP, a mask \( C \), called the composite mask, is considered, where \( c_{ij} = 1 \) if the \( a_{ij} \) and \( b_{ij} \) were assigned with the \( \text{assemble} \) algorithm. Otherwise, \( c_{ij} = 0 \). The elements from \( A \) assigned in \( \text{assemble} \) are denoted with a \( \text{masked flow matrix} \ A^C \), where \( a^C_{ij} = a_{ij} \), if \( c_{ij} = 1 \) and \( a^C_{ij} \) is otherwise undefined. Similarly, the elements from \( B \) assigned with \( \text{assemble} \) are denoted with a \( \text{masked distance matrix} \ B^C \). The corresponding pseudo-code is given in Algorithm [II] and it is explained in Section [II-A]

The function \( \text{fillingUp} \) assigns the positions in cQAP’s matrices that were not assigned with \( \text{assemble} \). Thus, a position \((i, j)\) in \( A \) and \( B \) is assigned with \( \text{fillingUp} \) if and only if \( c_{ij} = 0 \). We consider also a outside region mask matrix \( O = -C \). The elements from \( A \) assigned in \( \text{fillingUp} \) are denoted with a masked matrix \( A^O \), and the elements from \( B \) assigned in \( \text{fillingUp} \) are denoted with \( B^O \). \( \text{fillingUp} \) ensures that this outside region has the identity permutation as the optimal solution. The corresponding pseudo-code is given in Algorithm [III] and it is described in Section [II-B].

Section [II-C] discusses some aspects of the generated QAP instances. Note that the resulting cQAP does not generally have a known local optima, finding these conditions is outside the scope of this paper.

A. Aggregate QAPs

Let \( \mathcal{L} \) be an allocation strategy or function that assigns facilities from each component QAP, \((A^k, B^k, I)\) to the facilities
of the composite QAP. We denote with $j$ the facility from the cQAP that corresponds to the facility $i$ from the component $k$, $j ← L^k(i)$. To each facility of cQAP corresponds at least one facility of a component QAP. To each facility in a component QAP is assigned a unique facility in cQAP. The union of all assignment functions is \( \cup_{k=1}^n L^k(.) = \{1, \ldots, N\} \). Note that overlapping between facilities of different component QAPs are allowed. For simplicity, we consider that the facilities from component QAPs are adjacent in the composite QAP.

In Algorithm 2, for each pair of facilities \((i, j)\) in each \(k\)-th component QAP, there assigned a pair of facilities \((t, p)\) in the resulting \(A^c\). We update the values \(a_{tp} \in A^c\) and \(b_{tp} \in B^c\) with the corresponding values in \(a_{ij} \in A^k\) and \(b_{ij} \in B^k\). Then, \(a_{tp} ← a_{tp} + a_{ij}\) and \(b_{tp} ← b_{tp} + b_{ij}\), respectively. Since \(A^c\) and \(B^c\) should be symmetrical, we also set \(c_{tp} ← c_{tp}\) and \(c_{pt} ← c_{pt}\). The mask \(\mathcal{C}\) is updated to 1 for the pair of indices \(t\) and \(p\). \(c_{tp} ← 1\), and \(c_{pt} ← 1\). The returned, yet incomplete cQAP, is added to the output of Algorithm 3.

B. Filling up the cQAP

This algorithm guarantees that the elements in \(A^O\) and \(B^O\) obey the rearrangement inequality \([14]\). Informally, the rearrangement inequality states that the values in \(A^c\) and \(B^c\) are either larger or smaller than values in \(A^O\) and \(B^O\). Thus, the largest values in \(B^O\) correspond to the lowest values in \(A^O\), and the lowest values in \(B^O\) correspond to the largest values in \(A^O\).

Consider that the distribution \(D^A_H\) samples the lowest values of \(A_O\), and the distribution \(D^B_H\) samples the lowest values of \(B_O\). Consider that the distribution \(D^A_H\) samples the highest values of \(A_O\), and the distribution \(D^B_H\) samples the highest values of \(B_O\). We generate \(h\) values in \(A_O\) from \(D^A_H\) and \(1-h\) from \(D^A_L\). Because of the rearrangement inequality, \(h\) values in \(B_O\) are generated from \(D^B_H\) and \(1-h\) are generated from \(D^B_L\).

Algorithm 3 presents the pseudo-code for generating the elements in \(A_O\) and \(B_O\). fillingUp has as input parameters the set of distributions for the outside region, \(D^A_H, D^A_L, D^A_B, D^B_H, D^B_B\), and the percentage of values generated from \(D^A_H\) in \(A^O, h\).

Let \(S_A\) and \(S_B\) be ordered sets containing all the values from \(A^O\) and \(B^O\), respectively. Let \(r\) be the rank of \(a_{ij}\) in \(S_A\), where \(c_{ij} = 0\). If \(a_{ij}\) is generated from \(D^A_H\), then \(b_{ij}\) is generated from \(D^B_H\) such that the rank of \(b_{ij}\) in \(S_B\) is \(|S_A| - r\). Similarly, if \(a_{ij}\) is generated from \(D^B_H\), then \(b_{ij}\) is generated from \(D^B_H\) such that the rank of \(b_{ij}\) in \(S_B\) is \(|S_A| - r\).

C. Discussion

Additively decomposable cost function for cQAPs. Consider the set of all permutations of \(N\) facilities in the flow matrix. In permutation group theory, permutations are often written in cyclic form. If \(\pi\) is a permutation of facilities, we can write it as \(\pi = (\pi_1, \ldots, \pi_4)\), where \(\pi_k\) is a cycle containing a set of facilities that can be swapped with each other. These cycles are disjoint subsets. We consider \(d\) cycles, each cycle containing the facilities of exactly one component QAP. If there are \(n_k\) facilities in the \(k\)-th component QAP, the corresponding cycle is a \(n_k\)-cycle. The cycle of a component QAP \(k\) has the cost function

\[
c_k(\pi^k) = \sum_{i=1}^{n_k} \sum_{j=1}^{n_k} a_{ij} \cdot b_{\pi_i \pi_j}^k
\]

where \(k \in \{1, \ldots, d\}, d\) are the number of component QAPs, and \(\pi^k\) is a permutation of the facilities of the \(k\)-th component QAP, and \(\pi^k_i\) is the \(i\)-th value of the permutation \(\pi^k\). By design, the optimum cost for each cycle is \(c_k(\pi) = \min_{\pi^k} c_k(\pi^k)\).

The cost function of \(\pi\) on the cQAP is

\[
c(\pi) = \sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij} \cdot b_{\pi_i \pi_j} = \sum_{k=1}^{d} c_k(\pi^k)
\]

where \(Residue_C(\pi) = \sum_{c_{ij}=0} a_{ij} \cdot b_{\pi_i \pi_j}\) is a sum of costs for the elements in the outside region. \(Residue_C(\pi)\) is the sum over all overlapping positions of the overlapping component
QAPs and depends on the overlapping pattern. Thus, we consider that the proposed QAPs have additively decomposable cost functions with two residual terms representing costs of the overlapping parts and the outside region, respectively.

**Easy vs hard cases of QAP instances.** Cela [1] showed that QAP instances where all the elements obey the rearrangement inequality are easy. When the component QAPs are degenerated, \( n^1 = \ldots = n^d = 1 \), the cQAP becomes the “easy” QAP of Cela [1].

Let’s generate the elements in the component QAPs using uniform random distributions. According to Taillard [13], QAP instances generated with uniform random distributions are difficult because their global optimum is difficult to find. However, there are small differences in cost values between the global optimum and a local optimum. Furthermore, as the number of facilities goes to \( \infty \), this difference in cost function goes to 0.

We consider the composite region defined by \( C \) as the difficult region of the generated cQAP. The outside region defined by \( O \) is the easy region of the cQAP. By design, both the outside region and all the component QAPs are optimized by the identity permutation, but the composite QAP’s optimal solution is not, in general, the identity permutation.

**Parameter setting for cQAP instances.** Component QAPs are independently generated with a uniform random distribution, the same for all the components, \( D \). Let \( D^L \) and \( D^H \) be uniform independent distributions, and let the values of \( D^L \) and \( D^H \) be uniform randomly located in \( \mathcal{L}^O \) and \( \mathcal{B}^O \). Note that even though component QAPs and the elements in the outside region are generated with uniform random distributions, cQAP is not generated with a uniform random distribution.

To compare cQAP instances with the uniform randomly generated QAPs of Taillard [13], let the bounds of the uniform distributions that generate the flow and the distance matrices be the same. Consider the following setting: i) \( m = 20 \) and \( M = 30 \), ii) \( m^H = 90 \) and \( M^H = 99 \), and iii) \( m^L = 1 \) and \( M^L = 10 \). Let \( n = 8 \) be the number of facilities in component QAPs, where \( d \geq 2 \).

Let’s consider the percentage of high values in the outside region of the distance matrix, \( h \). We consider \( h \in (0.5, 1.0) \) to increase the variance in the outside region. This variance is larger than when there are only large or small elements in it, \( h = 0 \) or \( h = 1.0 \).

To compare our generator with Drezner et al. [2]’s generator, we also consider a certain amount of 0’s in \( D^L \); we denote them with \( z \). We consider three values for \( z \), \( z \in \{0, 0.5, 1.0\} \). \( z = 0 \) means that there are no 0’s in the outside region, whereas \( z = 1.0 \) means that there are only 0’s in \( D^L \). For the same \( h \), the variance of an outside region with lots of 0’s is smaller than the variance of an outside region with no 0’s.

**III. LANDSCAPE ANALYSIS WITH MULTI-RESTART LOCAL SEARCH**

The basin of attraction [3] of a local optimum, \( X \), is the set of independent uniformly distributed, iid, restarting points for local search of which the optimum is in \( X \). As performance indicator, we count the number of times the optimal solution is found. Section [III-A] shows that this definition of the basin of attraction is not informative even for small size QAPs because of the large number of basins of attraction in their landscape. In Section [III-B] we propose an alternative definition of the basin of attraction that reduce the number of basins of attraction by counting as one the local optima with the same value. In Section [III-C] we use empirical measures to approximate the size of basins of attraction.

We compare the basins of attraction of three types of QAPs: i) Taillard [13], ii) Drezner [2] and iii) our cQAPs with the parameters recommended in the previous section. All these three methods generate QAPs of various sizes using uniform randomly generated numbers distributed with some, simple or rather elaborated, strategy in the flow and distance matrices. Taillard’s QAPs have matrices with uniform randomly distributed elements. Drezner’s QAPs represent grids with uniform randomly generated points with a rather small amount of connections in the grid. The component QAPs and the elements in the outside region of cQAP are also uniformly randomly generated. Therefore, for all these QAPs, we assume a random configuration for basins of attraction.

### A. Basins of attraction

To approximate the number of basins of attraction, \( B \), we count the frequency of their occurrence. \( \beta_j \) is the number of local optima that are reached exactly \( j \) times. Like in [5], we assume that the distribution of \( \beta_j \) follows a family of parametrized distributions \( \text{Law}_{\gamma} \). The expected values \( \beta_{j, \gamma} = E_{\gamma}[\beta_j] \) are computed using

\[
\beta_{j, \gamma} = L \cdot \frac{\Gamma((j + \gamma) \cdot L \gamma)}{\Gamma((j + \gamma) \cdot \gamma)} \cdot \frac{M!}{(L \gamma + M)!} \cdot \frac{L!}{(M - j)!}
\]

where \( L \) is the number of basins of attraction visited so far, and \( M \) is the number of uniform randomly sampled solutions used to restart LS. To simplify this equation, we assume that \( \gamma \) is a positive integer. Then, \( \Gamma(\gamma) = \gamma! \). After re-grouping the terms of the above equation, we have

\[
\beta_{j, \gamma} = L \cdot \frac{M!}{(M - j)!} \cdot \left( \frac{1}{L + M} \right)^j
\]

where \( \left( \frac{M}{j} \right) = \frac{M!}{j!(M - j)!} \). \( \beta_{j, \gamma} \) are used to calculate \( B \), where \( \beta_{j, \gamma} \) is the solution of the equation

\[
\sum_{j=1}^{\infty} \beta_{j, \gamma} = 1 - (1 + M^{-1})^{-\gamma}
\]

Here, \( \gamma \in \{1, 2, 3\} \) is chosen to minimize the difference between \( \beta_j \) and \( \beta_{j, \gamma} \) in the \( \chi^2 \) test, \( \sum_{\gamma=1}^{\infty} (\beta_j - \beta_{j, \gamma})^2 / \beta_{j, \gamma} \). The probability that at least one point of the searching space lies in a basin of attraction is

\[
p_{B,M} \to L \to \infty \exp -a^{-1}
\]

where \( a \) is the solution of \( M = [L^2 \cdot a] \).

### Some experimental results

For this experiment, we restart the local search \( M = 10^5 \) times. In Figure [I] on the top,
we compare the estimated number of basins of attraction, $B$, versus the number of already visited basins of attraction $L$ for the three QAP instances. We set $M = 10^5$ and $\gamma = \{1, 2, 3\}$. For larger $N > 20$, for all the QAP instances, the values of estimated, $B$, and visited, $L$, basins of attraction are about the same $10^5$. The covering probability $p_{B,M}$ for the three QAPs is basically 0.

Thus, even for a small number of facilities, QAP instances are difficult with lots of local optima, where the restarting points are almost always located in another basin of attraction. We conclude that this definition of basin of attraction is not very informative for the prediction of the size and the number of local optima of QAPs.

### B. Fitness indecisive basins of attraction

The number of basins of attraction is reduced using the observation that LS cannot discriminate between solutions that have equal fitness values but different representations. These solutions, with the same value, can belong or not to the same plateau and, in general, they are difficult to differentiate and to store.

A fitness indecisive set of basin of attraction, in short indecisive basin of attraction, is the set of points for which the local optimal solutions obtained by restarting LS from these points have the same cost (or fitness value).

The number and the size of indecisive basins of attraction are computed with the same algorithm as the regular basins of attraction, with the logical difference that now the equally valued local optima are counted as one indecisive basin of attraction. We denote with $B_I$ the number of estimated indecisive basins of attraction, with $L_I$ the number of visited indecisive basins of attraction, and with $p_{B_I,M}$ the coverage probability of $B_I$.

### Experimental results

On the bottom of Figure 1 the number of approximated indecisive basins of attraction, $B_I$, versus the number of already visited indecisive basins of attraction $L_I$ for the three QAPs are compared. Note that even for the small size QAPs, cqap16 and tai15a, there are lots of indecisive basins of attraction, $L_I \approx 10^4$, which is about $M/10$, the coverage probability is again about 0. We deduce that $M = 10^5$ LS restarts are not enough for landscape analysis of composite and Taillard’s QAP instances.

Note that for small size Drezn’s QAP, dre15, the coverage probability $p_{B_I,M} = 0.6$ is large and $B_I$ is close to $L_I$ meaning that $M = 10^5$ is a reasonable number of restarts. Furthermore, dreN has non-zero coverage probabilities $p_{B_I,M} > 0.2$ even for medium size QAPs, $N \leq 56$, and the lowest $B_I$. Comparing these results with the results the standard basins of attraction, i.e. $B$, $L$ and $p_{B,M}$, we deduce that Drezn’s QAP has lots of basins of attraction with the same value of the local optimum, which can be grouped in fewer indecisive basins of attraction.

In conclusion, for the same number of facilities $N$, cqapN and taiNa are more difficult than dreN.

### C. Empirical analysis of basins of attraction

The number of the neighbourhood function calls in a LS, $\#N$, is an empirical approximation of the size of the basins of attraction. The number of swaps generated in a LS run is the number of neighbourhood calls multiplied with the number of solutions in a neighbourhood, $\#N \cdot \binom{N}{2}$. This measure gives an approximation of the time necessary for a LS to converge to a local optimum.

Figure 2(b) presents the average number of neighbourhood calls for the three types of QAPs. Note that all QAP instances with the same number of facilities have about the same number
of neighbourhood calls, and the number of visited solutions during a LS run, Figure 2 a), is about the same. In Figure 2 b), if \( N \) increases, then the number of times the neighbourhood function is called also increases. This is reflected also in Figure 2 a) where the number of visited solutions during a LS run increases when \( N \) increases.

The percentage of times LS’s local optimum is equal to the best value found in all runs from Figure 2 c) is a good indicator of the performance of the multi-restart LS algorithm. The identity permutation is a solution for \( dreN \) type of QAPs, whereas for Taillard’s QAPs, \( taiNa \) (or simpler \( taiN \)) the (near)optimal solutions are known from QAPLIB’s home-page. From \( M = 10^5 \) LS runs, for \( N > 12 \), the multi-restart LS finds almost never the solutions with best value. Unlike for \( cqapN \) and \( taiNa \), for all the instances \( dreN \), where \( N \leq 56 \), the multi-restart LS finds the optimal solution within \( M = 10^5 \) LS runs. As expected, the larger the number of facilities in \( dreN \), the lower the number of times multi-restart LS finds the best-so-far solution.

We conclude that the tested cQAPs are more difficult than Drezner’s QAP instances, which have lots of local optima with the same value.

IV. LANDSCAPE ANALYSIS WITH ITERATED LOCAL SEARCH

In this section, we show that the cQAP instances can be well explored with iterated LS [2], [3] because of the structure in the search space generated by the component QAPs. In Section IV-A, we investigate how to design an iterated LS algorithm that has a good balance between exploration vs exploitation using the method proposed in [3], [4], adapted for single objective search spaces. Section IV-B shows why iterated LS outperforms multi-restart LS on the cQAP instances.

A. Exploration/exploitation properties of iterated LS

The biggest advantage of multi-restart LS over iterated LS is that there are no parameters to set. For iterated LS, a mutation rate, or a set of mutation rates, needs to be set and the robustness of iterated LS’s behavior depends on the used (combinations of) mutations. In our analysis, we measure the exploitation and exploration characteristics of iterated LS - that are the success, improvement and escape probabilities - like in [4]. The escape probability is the probability that the solution after mutation does not belong to the basin of attraction of its parent

\[
p_{\text{escape}} = \frac{#\text{escapes}}{#\text{restarts}}
\]

where \#restarts are the number of times LS is restarted in a run. The success probability is the probability that the new starting solution escapes from the basin of attraction, and the local optimum generated by LS is performing at least as good as the current solution

\[
p_{\text{success}} = \frac{#\text{success}}{#\text{restarts}}
\]

The improvement probability is the probability that, if the restart escapes from the basin of attraction of its parent, the local optimum output by the restarted LS is at least as good as the current solution

\[
p_{\text{impro}} = \frac{#\text{success}}{#\text{escapes}}
\]

Note that \( p_{\text{impro}} = p_{\text{success}} / p_{\text{escape}} \).

All iterated LSs are run an equal number of swaps, 5412690, which is equivalent with restarting 100 LS from uniform randomly generated solutions on the same problem, \( cqap56 \). The results - that are escape, success and improvement probabilities are averaged over 1000 independent runs of iterated LS for each \( m = \{3, \ldots, 39\} \).
In the following, we use the iterated LS a mutation operator that uniform randomly selects from a range of exchange rates $m \in \{3, \ldots, \lfloor N/3 \rfloor\}$, where $N$ is the number of facilities of the cQAP as before.

Figure 4(a) shows that the escape probability of the iterated LS increases with the numbers of facilities. Note that the performance of iterated LS is inverse proportional with the escape probability, and very large exchange rates, e.g. larger than $\lfloor N/3 \rfloor$, decrease the performance of the algorithm.

In Figure 4(b), the success probabilities are within a small range between 0.02 and 0.04 and with large variances. We consider that iterated LS has a very different behaviour in different parts of the landscape, a larger number of LS restarts being required before this behaviour stabilizes.

In Figure 4(c), the improvement probabilities vary within a larger range than the success probabilities. The largest improvement is $\approx 0.12$ for $cqap20$ and the smallest improvement is $\approx 0.06$ for $cqap56$. The number of times LS is restarted for all iterated LS is about 200, meaning that, on average, there are at most 24 and at least 12 improvements per iterated LS run. The variance of the improvement probabilities is large, and our explanation for it is the small number of times iterated LS is restarted in a large search space.

Let’s now compare the results in Figure 5 and 4. Figure 4 shows that the iterated with $m \in \{3, \ldots, N/3\}$ has better balance between exploitation and exploration than all the iterated LSs that used a fixed $m$. In Figure 4(a), $p_{escape} = 0.8$ for $cqap56$ is approximatively equal with the escape probability of the iterated LS with a large exchange rate $m > 20$ from Figure 3(a). In Figure 5(b), $p_{success} = 0.04$ for $cqap56$ is larger that the largest success probability of the iterated LS from Figure 3(b). The corresponding $p_{improv} = 0.06$ in Figure 4(c) is larger than the improvement probability of an iterated LS with good exploration properties, $m > 5$, in Figure 5(c).

We conclude that randomly alternating exchange rates is beneficial for the success and improvement probabilities of the iterated LS, without decreasing the escape probability.

B. Exploring basins of attraction with iterated LS

The empirical analysis of the basins of attraction shows differences between multi-restart LS and iterated LS runs on cQAPs. In Figure 5(a), the mean number of solutions visited during a LS run is considerably larger for the multi-restart LS than for the iterated LS. Consequently, the number of neighbourhood calls per LS in Figure 5(b) for the iterated LS is considerably smaller than for the multi-restart LS. Because multi-restart and iterated LS run the same number of swaps and the number of visited solutions per LS is smaller for the iterated LS than for the multi-restart LS, for the same $cqapN$ instance, the number of times multi-restart LS is restarted in Figure 5(c) is much smaller than the number of times iterated LS is restarted. That means that, on average, iterated LS spends less time in a local search run than multi-restart LS does and, in turn, restarts LS more times.

Figure 5(c) shows that the iterated LS for $cqap56$ restarts LS approximately 200 times. The number of successes for the iterated LS is, on average, 12 that is considerably more than 1.8 for the iterated LS with $m = 20$. Figure 5(d) shows that

In Figure 3 we present the escape, success and improvement probabilities for $cqapN$ with $N \in \{12, \ldots, 56\}$. Each iterated LS generates restarting solutions using mutation and one of the exchange rates $m \in \{3, \ldots, N/3\}$ uniformly random chosen from this set: a) the escape, b) the success, c) the improvement probabilities.

Note that the behaviour of iterated LS greatly depends on the exchange rate of the mutation operator. For very large the restarted solutions are basically uniform randomly generated whereas for small $m$ the solutions might not even escape the basin of attraction. In the following paragraph, we show that using a set of exchange rates for the mutation operator rather than a single exchange rate is beneficial for the exploitation/exploitation properties of the iterated LS.
the iterated LS finds with probability 0.1 from 100 runs - that is 10 times - the best local optimum unlike multi-restart LS that cannot find this optimum at all.

We conclude that the iterated LS is performing better than the multi-restart LS because it exploits the structure of the search space. This difference in performance is especially large for medium size cQAPs. For the efficient exploration of cQAPs with large number of facilities, the iterated LS should be enhanced with other techniques, like population and recombination operators, but such a study is beyond the scope of this paper.

V. CONCLUSION

We propose an instance generator for composite QAPs (cQAPs) what combines a number of small size QAPs, whose exact solutions can be easily computed in order to create large QAP instance with lots of local optima solution and, the the same time, structure in the search space. The rest of the elements from the cQAP, which were not assigned with elements of component QAPs are generated using the rearrangement inequality. Note that both composite and outside regions have the identity permutation as the optimal solution. The number of times a solution with the best value is found in an iterated LS and multi-restart LS run.

We conclude that cQAPs are difficult QAP instances that are especially useful for testing heuristics which exploit the structure of the search space.

REFERENCES