Local Search Heuristics for the Multidimensional Assignment Problem[*]

Gregory Gutin[†] Daniel Karapetyan[‡]

Abstract

The Multidimensional Assignment Problem (MAP) (abbreviated s-AP in the case of s dimensions) is an extension of the well-known assignment problem. The most studied case of MAP is 3-AP, though the problems with larger values of s also have a large number of applications. We consider several known neighborhoods, generalize them and propose some new ones. The heuristics are evaluated both theoretically and experimentally and dominating algorithms are selected. We also demonstrate a combination of two neighborhoods may yield a heuristics which is superior to both of its components.

Keywords: Multidimensional Assignment Problem; Local Search; Neighborhood; Metaheuristics.

1 Introduction

The *Multidimensional Assignment Problem* (MAP) (abbreviated s-AP in the case of s dimensions, also called *(axial) Multi Index Assignment Problem*, MIAP, [5, 29]) is a well-known optimization problem. It is an extension of the *Assignment Problem* (AP), which is exactly the two dimensional case of MAP. While AP can be solved in the polynomial time [25], s-AP for every $s \geq 3$ is NP-hard [13] and inapproximable [9][1]. The most studied case of MAP is the case of three dimensions [1, 3, 4, 11, 20, 37] though the problem has a host of applications for higher numbers of dimensions, e.g., in matching information from several sensors (data association problem), which arises in plane tracking [27, 30], computer vision [39] and some others [3, 5, 7], in routing in meshes [5], tracking elementary particles [33], solving systems of polynomial equations [6], image recognition [14], resource allocation [14], etc.

For a fixed $s \geq 2$, the problem s-AP is stated as follows. Let $X_1 = X_2 = \ldots = X_s = \{1, 2, \ldots, n\}$. We will consider only vectors that belong to the Cartesian product $X = X_1 \times X_2 \times \ldots \times X_s$. Each vector $e \in X$ is assigned a non-negative weight $w(e)$. For a vector $e \in X$, the component e_j denotes its jth coordinate, i.e., $e_j \in X_j$. A collection A of $t \leq n$ vectors $A^1, A^2, \ldots, A^t$ is a *(feasible) partial assignment* if $A^i_j \neq A^k_j$ holds for each $i \neq k$ and $j \in \{1, 2, \ldots, s\}$. The *weight* of a partial assignment A is $w(A) = \sum_{i=1}^{t} w(A^i)$. An *assignment* (or *full assignment*) is a partial assignment with n vectors. The objective of s-AP is to find an assignment of minimal weight.

[*]A preliminary version of this paper was published in Golumbic Festschrift, volume 5420 of Lect. Notes Comput. Sci., pages 100–115, Springer, 2009.

[†]Royal Holloway, University of London, `gutin@cs.rhul.ac.uk`

[‡]Royal Holloway, University of London, `daniel.karapetyan@gmail.com`

[1] Burkard et al. show it for a special case of 3-AP and since 3-AP is a special case of s-AP the result can be extended to the general MAP

We also provide a *permutation form* of the assignment which is sometimes more convenient. Let $\pi_1, \pi_2, \ldots, \pi_s$ be permutations of $X_1, X_2, \ldots, X_s$ respectively. Then $\pi_1 \pi_2 \ldots \pi_s$ is an assignment with the weight $\sum_{i=1}^{n} w(\pi_1(i)\pi_2(i)\ldots\pi_s(i))$.

It is obvious that one permutation, say the first one, may be fixed without any loss of generality: $\pi_1 = 1_n$, where 1_n is the identity permutation of size n. Then the objective of the problem is as follows:

$$\min_{\pi_2,\ldots,\pi_s} \sum_{i=1}^{n} w(i\pi_1(i)\ldots\pi_s(i)) \ .$$

A graph formulation of the problem is as follows. Having an s-partite graph G with parts X_1, X_2, ..., X_s, where $|X_i| = n$, find a set of n disjoint cliques in G of the minimal total weight if every clique e in G is assigned a weight $w(e)$.

Finally, an integer programming formulation of the problem is as follows.

$$\min \sum_{i_1 \in X_1, \ldots, i_s \in X_s} w(i_1 \ldots i_s) \cdot x_{i_1 \ldots i_s}$$

subject to

$$\sum_{i_2 \in X_2, \ldots, i_s \in X_s} x_{i_1 \ldots i_s} = 1 \qquad \forall i_1 \in X_1 \ ,$$
$$\ldots$$
$$\sum_{i_1 \in X_1, \ldots, i_{s-1} \in X_{s-1}} x_{i_1 \ldots i_s} = 1 \qquad \forall i_s \in X_s \ ,$$

where $x_{i_1 \ldots i_s} \in \{0, 1\}$ for all i_1, ..., i_s and $|X_1| = \ldots = |X_s| = n$.

Sometimes the problem is formulated in a more general way if $|X_1| = n_1$, $|X_2| = n_2$, ..., $|X_s| = n_s$ and the requirement $n_1 = n_2 = \ldots = n_s$ is omitted. However this case can be easily transformed into the problem described above by complementing the weight matrix to an $n \times n \times \ldots \times n$ matrix with zeros, where $n = \max_i n_i$.

The problem was studied by many researchers. Several special cases of the problem were intensively studied in the literature (see [26] and references there) and for few classes of instances polynomial time exact algorithms were found, see, e.g., [8, 9, 21]. In many cases MAP remains hard to solve [26, 36]. For example, if there are three sets of points of size n on a Euclidean plain and the objective is to find n triples of points, one from each set, such that the total circumference or area of the corresponding triangles is minimal, the corresponding 3-AP is still NP-hard [36]. The asymptotic properties of some special instance families are studied in [14].

As regards the solution methods, apart from exact and approximation algorithms [4, 11, 26, 31, 32], several heuristics including construction heuristics [4, 16, 23, 28], greedy randomized adaptive search procedures [1, 27, 28, 35], metaheuristics [10, 20] and parallel heuristics [28] are presented in the literature. Several local search procedures are proposed and discussed in [1, 4, 5, 9, 10, 20, 28, 35].

The difference between the construction heuristics and local search is sometimes crucial. While a construction heuristic generates a solution from scratch and, thus, has some solution quality limitation, local search is intended to improve an existing solution and, thus, can be used after a construction heuristic or as a part of a more sophisticated heuristic, so called metaheuristic.

The contribution of our paper is in collecting and generalizing all local search heuristics known from the literature, proposing some new ones and detailed theoretical and evaluating them both theoretically and experimentally. For the purpose of experimental evaluation we also thoroughly discuss, classify the existing instance families and propose some new ones.

In this paper we consider only the general case of MAP and, thus, all the heuristics which rely on the special structures of the weight matrix are not included in the comparison. We also assume that the number of dimensions s is a small fixed constant while the size n can be arbitrary large.

2 Heuristics

In this section we discuss some well known and some new MAP local search heuristics as well as their combinations.

2.1 Dimensionwise Variations Heuristics

The heuristics of this group were first introduced by Bandelt et al. [5] for MAP with decomposable costs. However, having a very large neighborhood (see below), they are very efficient even in the general case. The fact that this approach was also used by Huang and Lim as a local search procedure for their memetic algorithm [20] confirms its efficiency.

The idea of the dimensionwise variation heuristics is as follows. Consider the initial assignment A in the permutation form $A = \pi_1 \pi_2 \dots \pi_s$ (see Section 1). Let $p(A, \rho_1, \rho_2, \dots, \rho_s)$ be an assignment obtained from A by applying the permutations $\rho_1, \rho_2, \dots, \rho_s$ to $\pi_1, \pi_2, \dots, \pi_s$ respectively:

$$p(A, \rho_1, \rho_2, \dots, \rho_s) = \rho_1(\pi_1) \rho_2(\pi_2) \dots \rho_s(\pi_s) \ . \qquad (1)$$

Let $p_D(A, \rho)$ be an assignment $p(A, \rho_1, \rho_2, \dots, \rho_s)$, where $\rho_j = \rho$ if $j \in D$ and $\rho_j = 1_n$ otherwise (1_n is the identity permutation of size n):

$$p_D(A, \rho) = p\left(A, \begin{cases} \rho & \text{if } 1 \in D \\ 1_n & \text{otherwise} \end{cases}, \begin{cases} \rho & \text{if } 2 \in D \\ 1_n & \text{otherwise} \end{cases}, \dots, \begin{cases} \rho & \text{if } s \in D \\ 1_n & \text{otherwise} \end{cases}\right) \ . \qquad (2)$$

On every iteration, the heuristic selects some nonempty set $D \subsetneq \{1, 2, \dots, s\}$ of dimensions and searches for a permutation ρ such that $w(p_D(A, \rho))$ is minimized.

For every subset of dimensions D, there are $n!$ different permutations ρ but the optimal one can be found in the polynomial time. Let $swap(u, v, D)$ be a vector which is equal to vector u in all dimensions $j \in \{1, 2, \dots, s\} \setminus D$ and equal to vector v in all dimensions $j \in D$:

$$swap(u, v, D)_j = \begin{cases} u_j & \text{if } j \notin D \\ v_j & \text{if } j \in D \end{cases} \qquad \text{for } j = 1, 2, \dots, s. \qquad (3)$$

Let matrix $[M_{i,j}]_{n \times n}$ be constructed as

$$M_{i,j} = w(swap(A^i, A^j, D)) \ . \qquad (4)$$

It is clear that the solution of the corresponding 2-AP is exactly the required permutation ρ. Indeed, assume there exists some permutation ρ' such that $w(p_D(A, \rho')) < w(p_D(A, \rho))$. Observe that $p_D(A, \rho) = \{swap(A^i, A^{\rho(i)}, D) : \ i \in \{1, 2, \dots, n\}\}$. Then we have

$$\sum_{i=1}^{n} w(swap(A^i, A^{\rho'(i)}, D)) < \sum_{i=1}^{n} w(swap(A^i, A^{\rho(i)}, D)) \ .$$

Since $w(swap(A^i, A^{\rho(i)}, D)) = M_{i,\rho(i)}$, the sum $\sum_{i=1}^{n} w(swap(A^i, A^{\rho(i)}, D))$ is already minimized to the optimum and no ρ' can exist.

The neighborhood of a dimensionwise heuristic is as follows:

$$N_{\text{DV}}(A) = \left\{ p_D(A, \rho) : \ D \in \mathcal{D} \text{ and } \rho \text{ is a permutation} \right\} , \tag{5}$$

where $\mathcal{D}$ includes all dimension subsets acceptable by a certain heuristic. Observe that

$$p_D(A, \rho) = p_{\overline{D}}(A, \rho^{-1}) , \tag{6}$$

where $\rho^{-1}(\rho) = \rho(\rho^{-1}) = 1_s$ and $\overline{D} = \{1, 2, \dots, s\} \setminus D$, and, hence,

$$\left\{ p_D(A, \rho) : \ \rho \text{ is a permutation} \right\} = \left\{ p_{\overline{D}}(A, \rho) : \ \rho \text{ is a permutation} \right\} \tag{7}$$

for any D. From (7) and the obvious fact that $p_{\varnothing}(A, \rho) = p_{\{1,2,\dots,s\}}(A, \rho) = A$ for any ρ we introduce the following restrictions for $\mathcal{D}$:

$$D \in \mathcal{D} \Rightarrow \overline{D} \notin \mathcal{D} \quad \text{ and } \quad \varnothing, \{1, 2, \dots, s\} \notin \mathcal{D} . \tag{8}$$

With these restrictions, one can see that for any pair of distinct sets $D_1, D_2 \in \mathcal{D}$ the equation $p_{D_1}(A, \rho_1) = p_{D_2}(A, \rho_2)$ holds if and only if $\rho_1 = \rho_2 = 1_n$. Hence, the size of the neighborhood $N_{\text{DV}}(A)$ is

$$|N_{\text{DV}}(A)| = |\mathcal{D}| \cdot (n! - 1) + 1 . \tag{9}$$

In [5] it is decided that the number of iterations should not be exponential with regards to neither n nor s while the size of the maximum $\mathcal{D}$ is $|\mathcal{D}| = 2^{s-1} - 1$. Therefore two heuristics, LS1 and LS2, are evaluated in [5]. LS1 includes only singleton values of D, i.e., $\mathcal{D} = \{D : \ |D| = 1\}$; LS2 includes only doubleton values of D, i.e., $\mathcal{D} = \{D : \ |D| = 2\}$. It is surprising but according to both [5] and our computational experience, the heuristic LS2 produces worse solutions than LS1 though it obviously has larger neighborhood and larger running times. We improve the heuristic by allowing $|D| \le 2$, i.e., $\mathcal{D} = \{D : \ |D| \le 2\}$. This does not change the theoretical time complexity of the algorithm but improves its performance. The heuristic LS1 is called 1DV in our paper; LS2 with $|D| \le 2$ is called 2DV. We also assume (see Section 1) that the value of s is a small fixed constant and, thus, introduce a heuristic sDV which enumerates all feasible (recall (8)) $D \subset \{1, 2, \dots, s\}$.

The order in which the heuristics take the values $D \in \mathcal{D}$ in our implementations is as follows. For 1DV it is $\{1\}, \{2\}, \dots, \{s\}$. 2DV begins as 1DV and then takes all pairs of dimensions: $\{1, 2\}, \{1, 3\}, \dots, \{1, s\}$, $\{2, 3\}, \dots, \{s - 1, s\}$. Note that because of (8) it enumerates no pairs of vectors for $s = 3$, and for $s = 4$ it only takes the following pairs: $\{2, 3\}$, $\{2, 4\}$ and $\{3, 4\}$. sDV takes first all sets D of size 1, then all sets D of size 2 and so on up to $|D| = \lfloor s/2 \rfloor$. If s is even then we should take only half of the sets D of size $s/2$ (recall (7)); for this purpose we take all the subsets of $D \subset \{2, 3, \dots, s\}$, $|D| = s/2$ in the similar order as before.

It is obvious that $N_{1\text{DV}}(A) \subseteq N_{2\text{DV}}(A) \subseteq N_{s\text{DV}}(A)$ for any s however for $s = 3$ all the neighborhoods are equal and for $s = 4$ 2DV and sDV also coincide.

According to (8) and (9), the neighborhood size of 1DV is

$$|N_{1\text{DV}}(A)| = s \cdot (n! - 1) + 1 ,$$

of 2DV is

$$|N_{2\text{DV}}(A)| = \begin{cases} (2^{s-1} - 1) \cdot (n! - 1) + 1 & \text{if } s \in \{3, 4\} \\ \left(\binom{s}{2} + s\right) \cdot (n! - 1) + 1 & \text{if } s \ge 5 \end{cases} ,$$

and of sDV is

$$|N_{s\text{DV}}(A)| = (2^{s-1} - 1) \cdot (n! - 1) + 1 .$$

The time complexity of every run of DV is $O(|\mathcal{D}| \cdot n^3)$ as every 2-AP takes $O(n^3)$ and, hence, the time complexity of 1DV is $O(s \cdot n^3)$, of 2DV is $O(s^2 \cdot n^3)$ and of MDV is $O(2^{s-1} \cdot n^3)$.

2.2 k-opt

The k-opt heuristic for 3-AP for $k = 2$ and $k = 3$ was first introduced by Balas and Saltzman [4] as a *pairwise* and *triple interchange heuristic*. 2-opt as well as its variations were also discussed in [1, 10, 27, 28, 31, 35] and some other papers. We generalize the heuristic for arbitrary values of k and s.

The heuristic proceeds as follows. For every subset of k vectors taken in the assignment A it removes all these vectors from A and inserts some new k vectors such that the assignment feasibility is preserved and its weight is minimized. Another definition is as follows: for every set of distinct vectors $e^1, e^2, \dots, e^k \in A$ let $X'_j = \{e^1_j, e^2_j, \dots, e^k_j\}$ for $j = 1, 2, \dots, k$. Let $A' = \{e'^1, e'^2, \dots, e'^k\}$ be the solution of this s-AP of size k. Replace the vectors $e^1, e^2, \dots, e^k$ in the initial assignment A with $e'^1, e'^2, \dots, e'^k$.

The time complexity of k-opt is obviously $O\left(\binom{n}{k} \cdot k!^{s-1}\right)$; for $k \ll n$ it can be replaced with $O(n^k \cdot k!^{s-1})$. It is a natural question if one can use some faster solver on every iteration. Indeed, according to Section 1 it is possible to solve s-AP of size k in $O(k!^{s-2} \cdot k^3)$. However, it is easy to see that $k!^{s-1} < k!^{s-2} \cdot k^3$ for every k up to 5, i.e., it is better to use the exhaustive search for any reasonable k. One can doubt that the exact algorithm actually takes $k!^{s-2} \cdot k^3$ operations but even for the lower bound $\Omega(k!^{s-2} \cdot k^2)$ the inequality $k!^{s-1} < k!^{s-2} \cdot k^2$ holds for any $k \le 3$, i.e., for all the values of k we actually consider.

Now let us find the neighborhood of the heuristic. For some set $\mathcal{I}$ and a subset $I \subset \mathcal{I}$ let a permutation ρ of elements in $\mathcal{I}$ be an *I-permutation* if $\rho(i) = i$ for every $i \in \mathcal{I} \setminus I$, i.e., if ρ does not move any elements except elements from I. Let $E = \{e^1, e^2, \dots, e^k\} \subset A$ be a set of k distinct vectors in A. For $j = 2, 3, \dots, s$ let ρ_j be an E_j-permutation, where $E_j = \{e^1_j, e^2_j, \dots, e^k_j\}$. Then a set $W(A, E)$ of all assignments which can be obtained from A by swapping coordinates of vectors E can be described as follows:

$$W(A, E) = \left\{p(A, 1_n, \rho_2, \rho_3, \dots, \rho_s) : \ \rho_j \text{ is an } E_j\text{-permutation for } j = 2, 3, \dots, s\right\} .$$

Recall that 1_n is the identity permutation of size n and $p(A, \rho_1, \rho_2, \dots, \rho_s)$ is defined by (1).

The neighborhood $N_{k\text{-opt}}(A)$ is defined as follows:

$$N_{k\text{-opt}}(A) = \bigcup_{E \subset A, |E| = k} W(A, E) . \tag{10}$$

Let $Y, Z \subset A$ such that $|Y| = |Z| = k$. Observe that $W(A, Y) \cap W(A, Z)$ is nonempty and apart from the initial assignment A this intersection may contain assignments which are modified only in the common vectors $Y \cap Z$. To calculate the size of the neighborhood of k-opt let us introduce $W'(A, E)$ as a set of all assignments in $W(A, E)$ such that every vector in E is modified in at least one dimension, where $E \subset A$ is the set of k selected vectors in the assignment A:

$$W'(A, E) = \left\{A' \in W(A, E) : \ |A \cap A'| = n - k\right\} .$$

Then the neighborhood $N_{k\text{-opt}}(A)$ of k-opt is

$$N_{k\text{-opt}}(A) = \bigcup_{E \subset A, |E| \le k} W'(A, E) \tag{11}$$

and since $W(A, Y) \cap W(A, Z) = \varnothing$ if $Y \ne Z$ we have

$$|N_{k\text{-opt}}(A)| = \sum_{E \subset A, |E| \le k} |W'(A, E)| = \sum_{i=0}^{k} \binom{n}{i} N^i , \tag{12}$$

where $N^i = |W(A, E)|$ for any E with $|E| = i$. Observe that

$$W'(A, E) = W(A, E) \setminus \bigcup_{E' \subsetneq E} W'(A, E')$$

and $|W(A, E)| = k!^{s-1}$ for $|E| = k$ and, hence,

$$N^k = k!^{s-1} - \sum_{i=0}^{k-1} \binom{k}{i} N^i \; . \tag{13}$$

It is obvious that $N^0 = 1$ since one can obtain exactly one assignment (the given one) by changing no vectors. From this and (13) we have $N^1 = 0$, $N^2 = 2^{s-1} - 1$ and $N^3 = 6^{s-1} - 3 \cdot 2^{s-1} + 2$. From this and (12) follows

$$|N_{\text{2-opt}}(A)| = 1 + \binom{n}{2}(2^{s-1} - 1) \; , \tag{14}$$

$$|N_{\text{3-opt}}(A)| = 1 + \binom{n}{2}(2^{s-1} - 1) + \binom{n}{3}(6^{s-1} - 3 \cdot 2^{s-1} + 2) \; . \tag{15}$$

In our implementation, we skip an iteration if the corresponding set of vectors E either consists of the vectors of the minimal weight ($w(e) = \min_{e \in X} w(e)$ for every $e \in E$) or all these vectors have remained unchanged during the previous run of k-opt.

It is assumed in the literature [4, 31, 35] that k-opt for $k > 2$ is too slow to be applied in practice. However, the neighborhood $N_{k\text{-opt}}$ do not only includes the neighborhood $N_{(k-1)\text{-opt}}$ but also grows exponentially with the growth of k and, thus, becomes very powerful. We decided to include 2-opt and 3-opt in our research. Greater values of k are not considered in this paper because of nonpractical time complexity (observe that the time complexity of 4-opt is $O(n^4 \cdot 24^{s-1})$) and even 3-opt with all the improvements described above still takes a lot of time to proceed. However, 3-opt is more robust when used in a combination with some other heuristic (see Section 2.4).

It is worth noting that our extension of the pairwise (triple) interchange heuristic [4] is not typical. Many papers [1, 10, 27, 31, 35] consider another neighborhood:

$$N_{k\text{-opt*}}(A) = \big\{p_D(A, \rho) : \; D \subset \{1, 2, \dots, s\}, |D| = 1 \text{ and } \rho \text{ moves at most } k \text{ elements}\big\} \; ,$$

where p_D is defined in (2). The size of such neighborhood is $|N_{k\text{-opt*}}(A)| = s \cdot \binom{n}{k} \cdot (k! - 1) + 1$ and the time complexity of one run of k-opt* in the assumption $k \ll n$ is $O(s \cdot n^k \cdot k!)$, i.e., unlike k-opt, it is not exponential with respect to the number of dimensions s which is considered to be important by many researchers. However, as it is stated in Section 1, we assume that s is a small fixed constant and, thus, the time complexity of k-opt is still reasonable. At the same time, observe that $N_{k\text{-opt*}}(A) \subset N_{\text{1-DV}}(A)$ for any $k \le n$, i.e., 1DV performs as good as n-opt* with the time complexity of 3-opt*. Only in the case of $k = 2$ the heuristic 2-opt* is faster in theory however it is known [7] that the expected time complexity of AP is significantly less than $O(n^3)$ and, thus, the running times of 2-opt* and 1DV are similar while 1DV is definitely more powerful. Because of this we do not consider 2-opt* in our comparison.

2.3 Variable Depth Interchange (v-opt)

The *Variable Depth Interchange* (VDI) was first introduced by Balas and Saltzman for 3-AP as a heuristic based on the well known Lin-Kernighan heuristic for the traveling salesman problem [4]. We provide here a natural extension v-opt of the VDI heuristic for the s-dimensional case, $s \geq 3$, and then improve this extension. Our computational experiments show that the improved version of v-opt is superior to the natural extension of VDI with respect to solution quality at the cost of a reasonable increase in running time. In what follows, v-opt refers to the improved version of the heuristic unless otherwise specified.

In [4], the heuristic is described quite briefly. Our contribution is not only in the extending, improving and analyzing it but also in a more detailed and, we believe, clearer explanation of it. We describe the heuristic in a different way to the description provided in [4], however, both versions of our algorithm are equal to VDI in case of $s = 3$. This fact was also checked by reproduction of the computational evaluation results reported in [4].

Further we will use function $U(u,v)$ which returns a set of swaps between vectors u and v. The difference between the two versions of v-opt is only in the $U(u,v)$ definition. For the natural extension of VDI, let $U(u,v)$ be a set of all the possible swaps (see (3)) in at most one dimension between the vectors u and v, where the coordinates in at most one dimension are swapped:

$$U(u,v) = \left\{ swap(u,v,D) : \ D \subset \{1,2,\dots,s\} \text{ and } |D| \leq 1 \right\} .$$

For the improved version of v-opt, let $U(u,v)$ be a set of all the possible swaps in at most $\lfloor s/2 \rfloor$ dimensions between the vectors v and w:

$$U(u,v) = \left\{ swap(u,v,D) : \ D \subset \{1,2,\dots,s\} \text{ and } |D| \leq s/2 \right\} .$$

The constraint $|D| \leq s/2$ guarantees that at least half of the coordinates of every swap are equal to the first vector coordinates. The computational experiments show that removing this constraint increases the running time and decreases the average solution quality.

Let vector $\mu(u,v)$ be the minimum weight swap between vectors u and v:

$$\mu(u,v) = \operatorname*{argmin}_{e \in U(u,v)} w(e) .$$

Let A be an initial assignment.

1. For every vector $c \in A$ do the rest of the algorithm.

2. Initialize the *total gain* $G = 0$, the *best assignment* $A_{\text{best}} = A$, and a set of available vectors $L = A \setminus \{c\}$.

3. Find vector $m \in L$ such that $w(\mu(c,m))$ is minimized. Set $v = \mu(c,m)$ and $\overline{v}_j = \{c_j, m_j\} \setminus \{v_j\}$ for every $1 \leq j \leq s$. Now $v \in U(c,m)$ is the minimum weight swap of c with some other vector m in the assignment, and $\overline{v}$ is the complementary vector.

4. Set $G = G + w(c) - w(v)$. If now $G \leq 0$, set $A = A_{\text{best}}$ and go to the next iteration (Step 1).

5. Mark m as an unavailable for the further swaps: $L = L \setminus \{m\}$. Note that c is already marked unavailable: $c \notin L$.

6. Replace m and c with v and $\overline{v}$. Set $c = \overline{v}$.

7. If $w(A) < w(A_{\text{best}})$, save the new assignment as the best one: $A_{\text{best}} = A$.

8. Repeat from Step 3 while the total gain is positive (see Step 4) and $L \neq \varnothing$.

The heuristic repeats until no improvement is found during a run. The time complexity of one run of v-opt is $O(n^3 \cdot 2^{s-1})$. The time complexity of the natural extension of VDI is $O(n^3 \cdot s)$, and the computation experiments also show a significant difference between the running times of the improved and the natural extensions. However, the solution quality of the natural extension for $s \geq 7$ is quite poor, while for the smaller values of s it produces solutions similar to or even worse than sDV solutions at the cost of much larger running times.

The neighborhood $N_{\text{v-opt}}(A)$ is not fixed and depends on the MAP instance and initial assignment A. The number of iterations (runs of Step 3) of the algorithm can vary from n to n^2. Moreover, there is no guarantee that the algorithm selects a better assignment even if the corresponding swap is in $U(c, m)$. Thus, we do not provide any results for the neighborhood of v-opt.

2.4 Combined Neighborhood

We have already presented two types of neighborhoods in this paper, let us say *dimensionwise* (Section 2.1) and *vectorwise* (Sections 2.2 and 2.3). The idea of the combined heuristic is to use the dimensionwise and the vectorwise neighborhoods togeather, combining them into so called Variable Neighborhood Search [38]. The combined heuristic improves the assignment by moving it into the local optimum with respect to the dimensionwise neighborhood, then it improves it by moving it to the local minimum with respect to the vectorwise neighborhood. The procedure is repeated until the assignment occurs in the local minimum with respect to both the dimensionwise and the vectorwise neighborhoods.

More formally, the combined heuristic DV_{opt} consists of a dimensionwise heuristic DV (either 1DV, 2DV or sDV) and a vectorwise heuristic *opt* (either 2-opt, 3-opt or v-opt). DV_{opt} proceeds as follows.

1. Apply the dimensionwise heuristic $A = DV(A)$.
2. Repeat:
   (a) Save the assignment weight $x = w(A)$ and apply the vectorwise heuristic $A = opt(A)$.
   (b) If $w(A) = x$ stop the algorithm.
   (c) Save the assignment weight $x = w(A)$ and apply the dimensionwise heuristic $A = DV(A)$.
   (d) If $w(A) = x$ stop the algorithm.

Step 1 of the combined heuristic is the hardest one. Indeed, it is typical that it takes a lot of iterations to move a bad solution to a local minimum while for a good solution it takes just a few iterations. Hence, the first of the two heuristics should be the most efficient one, i.e., it should perform quickly and produce a good solution. In this case the dimensionwise heuristics are more efficient because, having approximately the same as vectorwise heuristics time complexity, they search much larger neighborhoods. The fact that the dimensionwise heuristics are more efficient than the vectorwise ones is also confirmed by experimental evaluation (see Section 4).

It is clear that the neighborhood of a combined heuristic is defined as follows:

$$N_{\text{DV}_{\text{opt}}}(A) = N_{\text{DV}}(A) \cup N_{\text{opt}}(A) \ , \tag{16}$$

where $N_{\text{DV}}(A)$ and $N_{\text{opt}}(A)$ are neighborhoods of the corresponding dimensionwise and vectorwise heuristics respectively. To calculate the size of the neighborhood $N_{\text{DV}_{\text{opt}}}(A)$ we need to find the size of the intersection of these neighborhoods. Observe that

$$N_{\text{DV}}(A) \cap N_{k\text{-opt}}(A) = \left\{ p_D(A, \rho) : \ D \in \mathcal{D} \text{ and } \rho \text{ moves at most } k \text{ elements} \right\} \ , \tag{17}$$

where $p_D(A,\rho)$ is defined by (2). This means that, if r_k is the number of permutations on n elements which move at most k elements, the intersection (17) has size

$$|N_{\text{DV}}(A) \cap N_{k\text{-opt}}(A)| = |\mathcal{D}| \cdot (r_k - 1) + 1 \; . \tag{18}$$

The number r_k can be calculated as

$$r_k = \sum_{i=0}^{k} \binom{n}{i} \cdot d_i \; , \tag{19}$$

where d_i is the number of derangements on i elements, i.e., permutations on i elements such that none of the elements appear on their places; $d_i = i! \cdot \sum_{m=0}^{i} (-1)^m / m!$ [19]. For $k = 2$, $r_2 = 1 + \binom{n}{2}$; for $k = 3$, $r_3 = 1 + \binom{n}{2} + 2\binom{n}{3}$. From (9), (12), (16) and (18) we immediately have

$$\left|N_{\text{DV}_{k\text{-opt}}}(A)\right| = 1 + |\mathcal{D}| \cdot (n! - 1) + \left[\sum_{i=2}^{k} \binom{n}{i} N^i\right] - |\mathcal{D}| \cdot (r_k - 1) \; , \tag{20}$$

where N^i and r_k are calculated according to (13) and (19) respectively. Substituting the value of k, we have:

$$\left|N_{\text{DV}_{2\text{-opt}}}(A)\right| = 1 + |\mathcal{D}| \cdot (n! - 1) + \binom{n}{2}(2^{s-1} - 1) - |\mathcal{D}| \cdot \binom{n}{2} \qquad \text{and} \tag{21}$$

$$\begin{aligned}\left|N_{\text{DV}_{3\text{-opt}}}(A)\right| = 1 + |\mathcal{D}| \cdot (n! - 1) + \binom{n}{2}(2^{s-1} - 1) \\ + \binom{n}{3}(6^{s-1} - 3 \cdot 2^{s-1} + 2) - |\mathcal{D}| \cdot \left[\binom{n}{2} + 2\binom{n}{3}\right]\end{aligned} \tag{22}$$

One can easily substitute $|\mathcal{D}| = s$, $|\mathcal{D}| = \binom{s}{2}$ and $|\mathcal{D}| = 2^{s-1} - 1$ to (21) or (22) to get the neighborhood sizes of 1DV$_2$, 2DV$_2$, sDV$_2$, 1DV$_3$, 2DV$_3$ and sDV$_3$. We will only show the results for sDV$_2$:

$$\begin{aligned}|N_{s\text{DV}_2}(A)| &= 1 + (2^{s-1} - 1) \cdot (n! - 1) + \binom{n}{2}(2^{s-1} - 1) - (2^{s-1} - 1) \cdot \binom{n}{2} \\ &= 1 + (2^{s-1} - 1) \cdot (n! - 1) \; ,\end{aligned} \tag{23}$$

i.e., $|N_{s\text{DV}_2}(A)| = |N_{s\text{DV}}(A)|$. Since $N_{s\text{DV}}(A) \subseteq N_{s\text{DV}_2}(A)$ (see (16)), we can conclude that $N_{s\text{DV}_2}(A) = N_{s\text{DV}}(A)$. Indeed, the neighborhood of 2-opt can be defined as follows:

$$N_{2\text{-opt}} = \left\{p_D(A,\rho) : \; D \subset \{2, 3, \ldots, s\} \text{ and } \rho \text{ swaps at most two elements}\right\} ,$$

which is obviously a subset of $N_{s\text{DV}}(A)$ (see (5)). Hence, the combined heuristic sDV$_2$ is of no interest.

For other combinations the intersection (17) is significantly smaller than both neighborhoods $N_{\text{DV}}(A)$ and $N_{k\text{-opt}}(A)$ (recall that the neighborhood $N_{\text{v-opt}}$ has a variable structure). Indeed, $|N_{\text{DV}}(A)| \gg |N_{\text{DV}}(A) \cap N_{k\text{-opt}}(A)|$ because $|\mathcal{D}| \cdot (n! - 1) \gg |\mathcal{D}| \cdot (r_k - 1)$ for $k \ll n$. Similarly, $|N_{2\text{-opt}}(A)| \gg |N_{\text{DV}}(A) \cap N_{k\text{-opt}}(A)|$ because $\binom{n}{2}(2^{s-1} - 1) \gg |\mathcal{D}| \cdot \binom{n}{2}$ if $|\mathcal{D}| \ll 2^{s-1}$, which is the case for 1DV and 2DV if s is large enough. Finally, $|N_{3\text{-opt}}(A)| \gg |N_{\text{DV}}(A) \cap N_{k\text{-opt}}(A)|$ because $\binom{n}{2}(2^{s-1} - 1) + \binom{n}{3}(6^{s-1} - 3 \cdot 2^{s-1} + 2) \gg |\mathcal{D}| \cdot \left[\binom{n}{2} + 2\binom{n}{3}\right]$, which is true even for $|\mathcal{D}| = 2^{s-1}$, i.e., for sDV.

The time complexity of the combined heuristic is $O(n^k \cdot k!^{s-1} + |\mathcal{D}| \cdot n^3)$ in case of *opt* $= k$-opt and $O(n^3 \cdot (2^{s-1} + |\mathcal{D}|))$ if *opt* = v-opt. The particular formulas are provided in the following table:

	2-opt	3-opt	v-opt
1DV	$O(2^{s-1} \cdot n^2 + s \cdot n^3)$	$O(6^{s-1} \cdot n^3)$	$O(2^s \cdot n^3)$
2DV	$O(2^{s-1} \cdot n^2 + s^2 \cdot n^3)$	$O(6^{s-1} \cdot n^3)$	$O(2^s \cdot n^3)$
sDV	(no interest)	$O(6^{s-1} \cdot n^3)$	$O(2^s \cdot n^3)$

Note that all the combinations with 3-opt and v-opt have equal time complexities; this is because the time complexities of 3-opt and v-opt are dominant. Our experiments show that the actual running times of 3-opt and v-opt are really much higher then even the sDV running time. This means that the combinations of these heuristics with sDV are approximately as fast as the combinations of these heuristics with light dimensionwise heuristics 1DV and 2DV. Moreover, as it was noticed above in this section, the dimensionwise heuristic, being executed first, simplifies the job for the vectorwise heuristic and, hence, the increase of the dimensionwise heuristic power may decrease the running time of the whole combined heuristic. At the same time, the neighborhoods of the combinations with sDV are significantly larger than the neighborhoods of the combinations with 1DV and 2DV. We can conclude that the 'light' heuristics $1DV_3$, $2DV_3$, $1DV_v$ and $2DV_v$ are of no interest because the 'heavy' heuristics sDV_3 and sDV_v, having the same theoretical time complexity, are more powerful and, moreover, outperformed the 'light' heuristics in our experiments with respect to both solution quality and running time on average and in most of single experiments.

2.5 Other algorithms

Here we provide a list of some other MAP algorithms presented in the literature.

- A host of local search procedures and construction heuristics which often have some approximation guarantee ([5, 9, 11, 21, 26, 27] and some others) are proposed for special cases of MAP (usually with decomposable weights, see Section 3.2) and exploit the specifics of these instances. However, as it was stated in Section 1, we consider only the general case of MAP, i.e., all the algorithms included in this paper do not rely on any special structure of the weight matrix.

- A number of construction heuristics are intended to generate solutions for general case MAP [4, 16, 23, 28]. While some of them are fast and low quality, like Greedy, some, like Max-Regret, are significantly slower but produce much better solutions. A special class of construction heuristics, Greedy Randomized Adaptive Search Procedure (GRASP), was also investigated by many researchers [1, 27, 28, 35].

- Several metaheuristics, including a simulated annealing procedure [10] and a memetic algorithm [20], were proposed in the literature. Metaheuristics are sophisticated algorithms intended to search for the near optimal solutions in a reasonably large time. Proceeding for much longer than local search and being hard for theoretical analysis of the running time or the neighborhood, metaheuristics cannot be compared straightforwardly to local search procedures.

- Some weak variations of 2-opt are considered in [1, 27, 31, 35]. While our heuristic 2-opt tries all possible recombinations of a pair of assignment vectors, i.e., 2^{s-1} combinations, these variations only try the swaps in one dimension at a time, i.e., s combinations for every pair of vectors. We have already decided that these variations have no practical interest, for details see Section 2.2.

3 Test Bed

While the theoretical analysis can help in heuristic design, selection of the best approaches requires empirical evaluation [18, 34]. In this section we discuss the test bed and in Section 4 the experimental results are reported and discussed.

The question of selecting proper test bed is one of the most important questions in heuristic experimental evaluation [34]. While many researchers focused on instances with random independent weights ([3, 4, 24, 31] and some others) and random instances with predefined solutions [10, 15, 23], several more sophisticated models are of greater practical interest [5, 9, 11, 12, 26]. There is also a number of papers which consider real-world and pseudo real-world instances [6, 27, 30] but the authors of this paper suppose that these instances do not well represent all the instance classes and building a proper benchmark with the real-world instances is a subject for another research.

In this paper we group all the instance families into two classes: instances with independent weights (Section 3.1) and instances with decomposable weights (Section 3.2). Later we show that the heuristics perform differently on the instances of these classes and, thus, this devision helps us in correct experimental analysis of the local search algorithms.

3.1 Instances With Independent Weights

One of the most studied class of instances for MAP is *Random Instance Family*. In Random, the weight assigned to a vector is a random uniformly distributed integral value in the interval $[a, b-1]$. Random instances were used in [1, 3, 4, 32] and some others.

Since the instances are random and quite large, it is possible to estimate the average solution value for the Random Instance Family. The previous research in this area [24] show that if n tends to infinity than the problem solution approaches the bound an, i.e., the minimal possible assignment weight (observe that the minimal assignment includes n vectors of weight a). Moreover, an estimation of the mean optimal solution is provided in [14] but this estimation is not accurate enough for our experiments. In [18] we prove that it is very likely that every big enough Random instance has at least one an-assignment, where *x-assignment* means an assignment of weight x.

Let α be the number of assignments of weight an and let $c = b - a$. We would like to have an upper bound on the probability $\Pr(\alpha = 0)$. Such an upper bound is given in the following theorem whose proof (see [18]) is based on the Extended Jansen Inequality (see Theorem 8.1.2 of [2]).

Theorem 1. *For any n such that $n \geq 3$ and*

$$\left(\frac{n-1}{e}\right)^{s-1} \geq c \cdot 2^{\frac{1}{n-1}}, \tag{24}$$

we have $\Pr(\alpha = 0) \leq e^{-\frac{1}{2\sigma}}$, *where* $\sigma = \sum_{k=1}^{n-2} \frac{\binom{n}{k} \cdot c^k}{[n \cdot (n-1) \cdots (n-k+1)]^{s-1}}$.

The lower bounds of $\Pr(\alpha > 0)$ for different values of s and n and for $b - a = 100$, are reported below.

$s=4$		$s=5$		$s=6$		$s=7$	
n	$\Pr(\alpha > 0)$	n	$\Pr(\alpha > 0)$	n	$\Pr(\alpha > 0)$	n	$\Pr(\alpha > 0)$
15	0.575	10	0.991	8	1.000	7	1.000
20	0.823	11	0.998				
25	0.943	12	1.000				
30	0.986						
35	0.997						
40	1.000						

One can see that a 4-AP Random instance has an (an)-assignment with the probability which is very close to 1 if $n \geq 40$; a 5-AP instance has an (an)-assignment with probability very close to 1 for $n \geq 12$, etc.; so, the optimal solutions for all the Random instances used in our experiments (see Section 4) are very likely to be of weight an. For $s = 3$ Theorem 1 does not provide a good upper bound, but we are able to use the results from Table II in [4] instead. Balas and Saltzman report that in their experiments the average optimal solution of 3-AP for Random instances reduces very quickly and has a small value even for $n = 26$. Since the smallest Random instance we use in our experiments has size $n = 150$, we assume that all 3-AP Random instances considered in our experiment are very likely to have an an-assignment.

Useful results can also be obtained from (11) in [14] which is an upper bound for the average optimal solution. Grundel, Oliveira and Pardalos [14] consider the same instance family except the weights of the vectors are real numbers uniformly distributed in the interval $[a, b]$. However the results from [14] can be extended to our discrete case. Let $w'(e)$ be a real weight of the vector e in a continuous instance. Consider a discrete instance with $w(e) = \lfloor w'(e) \rfloor$ (if $w'(e) = b$, set $w(e) = b - 1$). Note that the weight $w(e)$ is a uniformly distributed integer in the interval $[a, b-1]$. The optimal assignment weight of this instance is not larger than the optimal assignment weight of the continuous instance and, thus, the upper bound for the average optimal solution for the discrete case is correct.

In fact, the upper bound $\bar{z}_u^*$ (see [14]) for the average optimal solution is not accurate enough. For example, $\bar{z}_u^* \approx an + 6.9$ for $s = 3$, $n = 100$ and $b - a = 100$, and $\bar{z}_u^* \approx an + 3.6$ for $s = 3$, $n = 200$ and $b - a = 100$, so it cannot be used for $s = 3$ in our experiments. The upper bound $\bar{z}_u^*$ gives a better approximation for larger values of s, e.g., $\bar{z}_u^* \approx an + 1.0$ for $s = 4$, $n = 40$ and $b - a = 100$, however, Theorem 1 provides stronger results ($\Pr(\alpha > 0) \approx 1.000$ for this case).

Another class of instances with almost independent weights is *GP Instance Family* which contains pseudo-random instances with predefined optimal solutions. GP instances are generated by an algorithm produced by Grundel and Pardalos [15]. The generator is naturally designed for s-AP for arbitrary large values of s and n. However, it is relatively slow and, thus, it was impossible to generate large GP instances. Nevertheless, this is what we need since finally we have both small (GP) and large (Random) instances with independent weights with known optimal solutions.

3.2 Instances With Decomposable Weights

In many cases it is not easy to define a weight for an s-tuple of objects but it is possible to define a relation between every pair of objects from different sets. In this case one should use *decomposable weights* [37] (or *decomposable costs*), i.e., the weight of a vector e should be defined as follows:

$$w(e) = f\left(d^{1,2}_{e_1,e_2}, d^{1,3}_{e_1,e_3}, \dots, d^{s-1,s}_{e_{s-1},e_s}\right) , \tag{25}$$

where $d^{i,j}$ is a distance matrix between the sets X_i and X_j and f is some function.

The most natural instance family with decomposable weights is Clique, which defines the function f as the sum of all arguments:

$$w_{\mathrm{c}}(e) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d^{i,j}_{e_i,e_j} . \tag{26}$$

The Clique instance family was investigated in [5, 11, 12] and some others. It was proven [11] that MAP restricted to Clique instances remains NP-hard.

A special case of Clique is *Geometric Instance Family*. In Geometric, the sets X_1, X_2, $\dots$, X_s correspond to sets of points in Euclidean space, and the distance between two points $u \in X_i$ and $v \in X_j$ is defined as Euclidean distance; we consider the two dimensional Euclidean space:

$$d_{\text{g}}(u, v) = \sqrt{(u_x - v_x)^2 + (u_y - v_y)^2} \,.$$

It is proven [36] that the Geometric instances are NP-hard to solve for $s = 3$ and, thus, Geometric is NP-hard for every $s \geq 3$.

In this paper, we propose a new special case of the decomposable weights, SquareRoot. It is a modification of the Clique instance family. Assume we have s radars and n planes and each radar observes all the planes. The problem is to assign signals which come from different radars to each other. It is quite natural to define a distance function between each pair of signals from different radars, and for a set of signals which correspond to one plane the sum of the distances should be small so (26) is a good choice. However, it is not actually correct to minimize the total distance between the assigning signals but one should also ensure that none of these distances is too large. Same requirements appear in a number of other applications. We propose a weight function which can leads to both small total distance between the assigned signals and small dispersion of the distances:

$$w_{\text{sq}}(e) = \sqrt{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(d^{i,j}_{e_i, e_j} \right)^2} \,. \tag{27}$$

Similar approach is used in [26] though they do not use square root, i.e., a vector weight is just a sum of squares of the edge weights in a clique. In addition, the edge weights in [26] are calculated as distances between some nodes in a Euclidean space.

Another special case of the decomposable weights, Product, is studied in [9]. Burkard et al. consider 3-AP and define the weight $w(e)$ as $w(e) = a^1_{e_1} \cdot a^2_{e_2} \cdot a^3_{e_3}$, where a^1, a^2 and a^3 are random vectors of positive numbers. It is easy to show that the Product weight function can be represented in the form (25). It is proven that the minimization problem for the Product instances is NP-hard in case $s = 3$ and, thus, it is NP-hard for every $s \geq 3$.

4 Computational Experimentation

In this section, the results of empirical evaluation are reported and discussed. The experiments were conducted for the following instances (for instance family definitions see Section 3):

- Random instances where each weight was randomly chosen in $\{1, 2, \ldots, 100\}$, i.e., $a = 1$ and $b = 101$. According to Subsection 3.1, the optimal solutions of all the considered Random instances are very likely to be $an = n$.
- GP instances with predefined optimal solutions (see Section 3.1).
- Clique and SquareRoot instances, where the weight of each edge in the graph was randomly selected from $\{1, 2, \ldots, 100\}$. Instead of the optimal solution value we use the best known solution value.
- Geometric instances, where both coordinates of every point were randomly selected from $\{1, 2, \ldots, 100\}$. The distances between the points are calculated precisely while the weight of a vector is rounded to the nearest integer. Instead of the optimal solution value we use the best known solution value.
- Product instances, where every value a^j_i was randomly selected from $\{1, 2, \ldots, 10\}$. Instead of the optimal solution value we use the best known solution value.

An instance name consists of three parts: the number s of dimensions, the type of the instance ('gp' for GP, 'r' for Random, 'c' for Clique, 'g' for Geometric, 'p' for Product and 'sr' for SquareRoot), and the size n of the instance. For example, `5r40` means a five dimensional Random instance of size 40. For every combination of instance size and type we generated 10 instances, using the number $seed = s + n + i$ as a seed of the random number sequences, where i is an index of the instance of this type and size, $i \in \{1, 2, \ldots, 10\}$. Thereby, every experiment is conducted for 10 different instances of some fixed type and size, i.e., every number reported in the tables below is average for 10 runs for 10 different instances.

The sizes of all but GP instances are selected such that every algorithm could process them all in approximately the same time. The GP instances are included in order to examine the behavior of the heuristics on smaller instances (recall that GP is the only instance set for which we know the exact solutions for small instances).

All the heuristics are implemented in Visual C++. The evaluation platform is based on AMD Athlon 64 X2 3.0 GHz processor.

Further, the results of the experiments of three different types are provided and discussed:

- In Subsection 4.1, the local search heuristics are applied to the assignments generated by some construction heuristic. These experiments allow us to exclude several local searches from the rest of the experiments, however, the comparison of the results is complicated because of the significant difference in both the solution quality and the running time.
- In Subsection 4.2, two simple metaheuristics are used to equate the running times of different heuristics. This is done by varying of number of iterations of the metaheuristics.
- In Subsection 4.3, the results of all the discussed approaches are gathered in two tables to find the most successful solvers for the instance with independent and decomposable weights for every particular running time.

4.1 Pure Local Search Experiments

First, we run every local search heuristic for every instance exactly once. The local search is applied to solutions generated with one of the following construction heuristics:

1. Trivial, which was first mentioned in [4] as *Diagonal*. Trivial construction heuristic simply assigns $A^i_j = i$ for every $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, s$.
2. Greedy heuristic was discussed in many papers, see, e.g. [4, 9, 16, 17, 18, 23]. It was proven [16] that in the worst case Greedy produces the unique worst solution; however, it was shown [17] that in some cases Greedy may be a good selection as a fast and simple heuristic.
3. Max-Regret was discussed in a number of papers, see, e.g., [4, 9, 16, 23, 35]. As for Greedy, it is proven [16] that in the worst case Max-Regret produces the unique worst solution however many researchers [4, 23] noted that Max-Regret is quite powerful in practice.
4. ROM was first introduced in [16] as a heuristic of a large domination number. On every iteration, the heuristic calculates the total weight for every set of vectors with the fixed first two coordinates: $M_{i,j} = \sum_{e \in X, e_1 = i, e_2 = j} w(e)$. Then it solves a 2-AP for the weight matrix M and reorders the second dimension of the assignment according to this solution and the first dimension of the assignment. The procedure is repeated recursively for the subproblem where the first dimension is excluded. For details see [16, 23].

We will begin our discussion from the experiments started from trivial assignments. The results reported in Tables 2 and 3 are averages for 10 experiments since every row of these tables corresponds to 10 instances of some fixed type and size but of different seed values (see above). The tables are split into two parts; the first part contains only the instances with independent weights (GP and Random) while the second part contains only the instances with decomposable weights (Clique, Geometric, Product and SquareRoot). The average values for different instance families and numbers of dimensions are provided at the bottom of each part of each table. The tables are also split vertically according to the classes of heuristics. The winner in every row and every class of heuristics is underlined.

The value of the solution error is calculated as $\big(w(A)/w(A_{\text{best}}) - 1\big) \cdot 100\%$, where A is the obtained assignment and A_{best} is the optimal assignment (or the best known one, see above).

In the group of the vectorwise heuristics the most powerful one is definitely 3-opt. v-opt outperforms it only in a few experiments, mostly three dimensional ones (recall that the neighborhood of k-opt increases exponentially with the increase of the number of dimensions s). As it was expected, 2-opt never outperforms 3-opt since $N_{\text{2-opt}} \subset N_{\text{3-opt}}$ (see Section 2.2). The tendencies for the independent weight instances and for the decomposable weight instances are similar; the only difference which is worth to note is that all but v-opt heuristics of this group solve the Product instances very well. Note that the dispersion of the weights in Product instances is really high and, thus, v-opt, which minimizes the weight of only one vector in every pair of vectors while the weight of the complementary vector may increase arbitrary, cannot be efficient for them.

As one can expect, sDV is more successful than 2DV and 2DV is more successful than 1DV with respect to the solution quality (obviously, all the heuristics of this group perform equally for 3-AP and 2DV and sDV are also equal for 4-AP, see Section 2.1). However, for the instances with decomposable weights all the dimensionwise heuristics perform very similarly and even for the large s, sDV is not significantly more powerful than 1DV or 2DV which means that in case of decomposable instances the most efficient iterations are when $|D| = 1$. We can assume that if c is the number of edges connecting the fixed and unfixed parts of the clique, then an iteration of a dimensionwise heuristic is rather efficient when c is small. Observe that, e.g., for Clique the diversity of values in the weight matrix $[M_{i,j}]_{n \times n}$ (see (4)) decreases with the increase of the number c and, hence, the space for optimization on every iteration is decreasing. Observe also that in the case $c = 1$ the iteration leads to the optimal match between the fixed and unfixed parts of the assignment vectors.

All the combined heuristics show improvements in the solution quality over each of their components, i.e., over both corresponding vectorswise and dimensionwise local searches. In particular, 1DV$_2$ outperforms both 2-opt and 1DV, 2DV$_2$ outperforms both 2-opt and 2DV, sDV$_3$ outperforms both 3-opt and sDV and sDV$_\text{v}$ outperforms both v-opt and sDV. Moreover, sDV$_3$ is significantly faster than 3-opt and sDV$_\text{v}$ is significantly faster than v-opt. Hence, we will not discuss the single heuristics 3-opt and v-opt in the rest of the paper. The heuristics 1DV$_2$ and 2DV$_2$, obviously, perform equally for 3-AP instances.

While for the instances with independent weights the combination of the dimensionwise heuristics with the vectorwise ones significantly improves the solution quality, it is not the case for the instances with decomposable weights (observe that 1DV performs almost as well as the most powerful heuristic sDV$_3$) which shows the importance of the instances division. We conclude that the vectorwise neighborhoods are not efficient for the instances with decomposable weights.

Next we conducted the experiments starting from the other construction heuristics. But first we compared the construction heuristics themselves, see Table 1. It is not surprising that Trivial produces the worst solutions. However, one can see that Trivial outperforms Greedy and Max-Regret for every Product instance. The reason is in the extremely high dispersion of the weights in Product. Both Greedy and Max-Regret construct the assignments by adding new vectors to it. The decision which vector should be added does not depend (or does not depend enough in case of Max-Regret) on the rest of the vectors and, thus, at the end of the procedure only the vectors with huge weights are available. For other instance families, Greedy, Max-Regret and ROM perform similarly though the running time of the heuristics is very different. Max-Regret is definitely the slowest construction heuristic; Greedy is very fast for the Random instances (this is because of the large number of vectors of the weight a and the implementation features, see [23] for details) and relatively slow for the rest of the instances; ROM's running time almost does not depend on the instance and is constantly moderate.

Starting from Greedy (Table 4) significantly improves the solution quality. This mostly influenced the weakest heuristics, e.g., 2-opt average error decreased in our experiments from 59% and 20% to 15% and 6% for independent and decomposable weights respectively, though, e.g., the most powerful heuristic sDV$_3$ error also noticeably decreased (from 2.8% and 5.8% to 2.0% and 2.5%). As regards the running time, Greedy is slower than most of the local search heuristics and, thus, the running times of all but sDV$_3$ and sDV$_\text{v}$ heuristics are very similar. The best of the rest of the heuristics in this experiment is sDV though 1DV$_2$ and 2DV$_2$ perform similarly.

Starting from Max-Regret improves the solution quality even more but at the cost of very large running times. In this case the difference in the running time of the local search heuristics almost disappears and sDV$_3$, the best one, reaches the average error values 1.3% and 2.2% for independent and decomposable weights respectively. Starting from ROM improves the quality only for the worst heuristics. This is probably because all the best heuristics contain sDV which does a good vectorwise optimization (recall that ROM exploits a similar to the dimensionwise neighborhood idea). At the same time, starting from ROM increases the running time of the heuristics significantly; the results for both Max-Regret and ROM are excluded from the paper; one can find them on the web [22].

It is clear that the construction heuristics are quite slow comparing to the local search and we should answer the following question: is it worth to spend so much time on the initial solution construction or there is some way to apply local search several times in order to improve the assignments iteratively? It is known that the algorithms which apply local search several times are called metaheuristics. There is a number of different metaheuristic approaches such as tabu search or memetic algorithms, but this is not the subject of this paper. In what follows, we are going to use two simple metaheuristics, Chain and Multichain.

4.2 Experiments With Metaheuristics

It is obvious that there is no sense in applying a local search procedure to one solution several times because the local search moves the solution to a local minimum with respect to its neighborhood, i.e., the second exploration of this neighborhood is useless. In order to apply the local search several times, one should perturb the solution obtained on the previous iteration. This idea immediately brings us to the first metaheuristic, let us say Chain:

```
1. Initialize an assignment $A$;
2. Set $A_\text{best} = A$;
3. Repeat:
    (a) Apply local search $A = LS(A)$;
    (b) If $w(A) < w(A_\text{best})$ set $A_\text{best} = A$;
    (c) Perturb the assignment $A = Perturb(A)$.
```

In this algorithm we use two subroutines, $LS(A)$ and $Perturb(A)$. The first one is some local search procedure and the second one is an algorithm which removes the given assignment from the local minimum by a random perturbation of it. The perturbation should be strong enough such that the assignment will not come back to the previous position on the next iteration every time though it should not be too strong such that the results of the previous search would be totally destroyed. Our perturbation procedure selects $p = \lceil n/25 \rceil + 1$ vectors in the assignment and perturbs them randomly. In other words, $Perturb(A)$ is just a random move of the p-opt heuristic. The parameters of the procedure are obtained empirically.

One can doubt if Chain is good enough for large running times and, thus, we introduce a little bit more sophisticated metaheuristic, Multichain. Unlike Chain, Multichain maintains several assignments on every iteration:

1. Initialize assignment A_{best};
2. Set $P = \varnothing$ and repeat the following $c(c+1)/2$ times:
   $P = P \cup \{LS(Perturb(A_{\text{best}}))\}$
   (recall that $Perturb(A)$ produces a different assignment every time);
3. Repeat:
   (a) Save the best c assignments from P into $C_1, C_2, \dots, C_c$ such that $w(C_i) \le w(C_{i+1})$;
   (b) If $w(C_1) < w(C_{\text{best}})$ set $A_{\text{best}} = C_1$.
   (c) Set $P = \varnothing$ and for every $i = 1, 2, \dots, c$ repeat the following $c - i + 1$ times: $P = P \cup \{LS(Perturb(C_i))\}$.

The parameter c is responsible for the power of Multichain; we use $c = 5$ and, thus, the algorithm performs $c(c+1)/2 = 15$ local searches on every iteration.

The results of the experiments with Chain running for 5 and 10 seconds are provided in Tables 5 and 6 respectively. The experiments are repeated for three construction heuristics, Trivial, Greedy and ROM. It was not possible to include Max-Regret in the comparison because it takes much more than 10 seconds for some of the instances.

The diversity in solution quality of the heuristics decreased with the usage of a metaheuristic. This is because the fast heuristics are able to repeat more times than the slow ones. Note that sDV$_3$, which is the most powerful single heuristic, is now outperformed by other heuristics. The most successful heuristics for the instances with independent and decomposable weights are sDV$_v$ and 1DV respectfully, though 1DV$_2$ and 2DV$_2$ are slightly more successful than sDV$_v$ for the GP instances. This result also holds for Multichain, see Tables 7 and 8. The success of 1DV confirms again that a dimensionwise heuristic is most successful when $|D| = 1$ if the weights are decomposable and that it is more efficient to repeat these iterations many times rather than try $|D| > 1$. For the explanation of this phenomenon see Section 4.1. The success of 1DV$_2$ and 2DV$_2$ for GP means existence of a certain structure in the weight matrices of these instances.

One can see that the initialization of the assignment is not crucial for the final solution quality. However, using Greedy instead of Trivial clearly improves the solutions for almost every instance and local search heuristic. In contrast to Greedy, using of ROM usually does not improve the solution quality. It only influences 2-opt which is the only pure vectorwise local search in the comparison (recall that ROM has a dimensionwise structure and, thus, it is good in combination with vectorwise heuristics).

The Multichain metaheuristic, given the same time, obtains better results than Chain. However, Multichain fails for some combinations of slow local search and hard instance because it is not able to complete even the first iteration in the given time. Chain, having much easier iterations, do not have this disadvantage.

Giving more time to a metaheuristic also improves the solution quality. Therefore, one is able to obtain high quality solutions using metaheuristics with large running times.

4.3 Solvers Comparison

To compare all the heuristics and metaheuristics discussed in this paper we produced Tables 9 and 10. These tables indicate which heuristics should be chosen to solve particular instances in the given time limitations. Several best heuristics are selected for every combination of the instance and the given time. A heuristic is included in the table if it was able to solve the problem in the given time, and if its solution quality is not worse than $1.1 \cdot w(A_{\text{best}})$ and its running time is not larger than $1.1 \cdot t_{\text{best}}$, where A_{best} is the best assignment produced by the considered heuristics and t_{best} is the time spent to produce A_{best}.

The following information is provided for every solver in Tables 9 and 10:

- Metaheuristic type (**C** for Chain, **MC** for Multichain or empty if the experiment is single).
- Local search procedure (2-opt, 1DV, 2DV, sDV, 1DV$_2$, 2DV$_2$, sDV$_3$ sDV$_\text{v}$ or empty if no local search was applied to the initial solution).
- Construction heuristic the experiment was started with (Gr, M-R or empty if the assignment was initialized by Trivial).
- The solution error in percent.

The following solvers were included in this experiment:

- Construction heuristics Greedy, Max-Regret and ROM.
- Single heuristics 2-opt, 1DV, 2DV, sDV, 1DV$_2$, 2DV$_2$, sDV$_3$ and sDV$_\text{v}$started from either Trivial, Greedy, Max-Regret or ROM.
- Chain and Multichain metaheuristics for either 2-opt, 1DV, 2DV, sDV, 1DV$_2$, 2DV$_2$, sDV$_3$ or sDV$_\text{v}$ and started from either Trivial, Greedy, Max-Regret or ROM. The metaheuristics proceeded until the given time limitations.

Note that for certain instances we exclude duplicating solvers (recall that all the dimensionwise heuristics perform equally for 3-AP as well as 2DV and sDV perform equally for 4-AP, see Section 2.1). The common rule is that we leave sDV rather than 2DV and 2DV rather than 1DV. For example, if the list of successful solvers for some 3-AP instance contains **C** 1DV Gr, **C** 2DV Gr and **C** sDV Gr, then only **C** sDV Gr will be included in the table. This is also applicable to the combined heuristics, e.g, having 1DV$_2$ R and 2DV$_2$ R for a 3-AP instance, we include only 2DV$_2$ R in the final results.

The last row in every table indicates the heuristics which are the most successful on average, i.e., the heuristics which can solve all the instances with the best average results.

Single construction heuristics are not presented in the tables; single local search procedures appear only for the small allowed times when all other heuristics take more time to run; the most of the best solvers are the metaheuristics. Multichain seems to be more suitable than Chain for large running times; however, Multichain does not appear for the instances with small n. This is probably because the power of the perturbation degree increases with the decrease of the instance size (note that $perturb(A)$ perturbs at least two vectors in spite of n).

The most successful heuristics for the assignment initialization are Trivial and Greedy; Trivial is useful rather for small running times. Max-Regret and ROM appear only a few times in the tables.

The success of a local search depends on the instance type. The most successful local search heuristic for the instance with independent weights is definitely sDV$_\text{v}$. The sDV heuristic also appears several times in Table 9, especially for the small running times. For the instances with decomposable weights, the most successful are the dimensionwise heuristics and, in particular, 1DV.

5 Conclusions

Several neighborhoods are generalized and discussed in this paper. An efficient approach of joining different neighborhoods is successfully applied; the yielded heuristics showed that they combine the strengths of their components. The experimental evaluation for a set of instances of different types show that there are several superior heuristic approaches suitable for different kinds of instances and running times. Two kinds of instances are selected: instances with independent weights and instances with decomposable weights. The first ones are better solvable by a combined heuristic $s\mathsf{DV}_\mathsf{v}$; the second ones are better solvable by 1DV. In both cases, it is good to initialize the assignment with the Greedy construction heuristic if there is enough time; otherwise one should use a trivial assignment as the initial one. The results can also be significantly improved by applying metaheuristic approaches for as log as possible.

Thereby, it is shown in the paper that metaheuristics applied to the fast heuristics dominate slow heuristics and, thus, further research of some more sophisticated metaheuristics such as memetic algorithms is of interest.

References

[1] R. M. Aiex, M. G. C. Resende, P. M. Pardalos, and G. Toraldo. Grasp with path relinking for three-index assignment. *INFORMS J. on Computing*, 17(2):224–247, 2005.

[2] N. Alon and J. Spencer. *The Probabilistic Method*. John Wiley, second edition, 2000.

[3] S. M. Andrijich and L. Caccetta. Solving the multisensor data association problem. *Nonlinear Analysis*, 47(8):5525–5536, 2001.

[4] E. Balas and M. J. Saltzman. An algorithm for the three-index assignment problem. *Oper. Res.*, 39(1):150–161, 1991.

[5] H. J. Bandelt, A. Maas, and F. C. R. Spieksma. Local search heuristics for multi-index assignment problems with decomposable costs. *Journal of the Operational Research Society*, 55(7):694–704, 2004.

[6] H. Bekker, E. P. Braad, and B. Goldengorin. Using bipartite and multidimensional matching to select the roots of a system of polynomial equations. In *Computational Science and Its Applications — ICCSA 2005*, volume 3483 of *Lecture Notes Comp. Sci.*, pages 397–406. Springer, 2005.

[7] R. E. Burkard and E. Çela. Linear assignment problems and extensions. In Z. Du and P. Pardalos, editors, *Handbook of Combinatorial Optimization*, pages 75–149. Dordrecht, 1999.

[8] R. E. Burkard, B. Klinz, and R. Rudolf. Perspectives of monge properties in optimization. *Discrete Applied Mathematics*, 70(2):95–161, 1996.

[9] R. E. Burkard, R. Rudolf, and G. J. Woeginger. Three-dimensional axial assignment problems with decomposable cost coefficients. Technical Report 238, Graz, 1996.

[10] W. K. Clemons, D. A. Grundel, and D. E. Jeffcoat. *Theory and algorithms for cooperative systems*, chapter Applying simulated annealing to the multidimensional assignment problem, pages 45–61. World Scientific, 2004.

[11] Y. Crama and F. C. R. Spieksma. Approximation algorithms for three-dimensional assignment problems with triangle inequalities. *European Journal of Operational Research*, 60(3):273–279, 1992.

[12] A. M. Frieze and J. Yadegar. An algorithm for solving 3-dimensional assignment problems with application to scheduling a teaching practice. *Journal of the Operational Research Society*, 32:989–995, 1981.

[13] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, January 1979.

[14] D. Grundel, C. Oliveira, and P. Pardalos. Asymptotic properties of random multidimensional assignment problems. *Journal of Optimization Theory and Applications*, 122(3):33–46, 2004.

[15] D. A. Grundel and P. M. Pardalos. Test problem generator for the multidimensional assignment problem. *Comput. Optim. Appl.*, 30(2):133–146, 2005.

[16] G. Gutin, B. Goldengorin, and J. Huang. Worst case analysis of max-regret, greedy and other heuristics for multidimensional assignment and traveling salesman problems. *Journal of Heuristics*, 14(2):169–181, 2008.

[17] G. Gutin and D. Karapetyan. Greedy like algorithms for the traveling salesman problem and multidimensional assignment problem. In *Advances in Greedy Algorithms*. I-Tech, 2008.

[18] G. Gutin and D. Karapetyan. A selection of useful theoretical tools for the design and analysis of optimization heuristics. *Memetic Computing*, 1(1):25–34, 2009.

[19] J. M. Harris, J. L. Hirst, and M. J. Mossinghoff. *Combinatorics and Graph Theory*. Mathematics, 2008.

[20] G. Huang and A. Lim. A hybrid genetic algorithm for the three-index assignment problem. *European Journal of Operational Research*, 172(1):249–257, July 2006.

[21] V. Isler, S. Khanna, J. Spletzer, and C. J. Taylor. Target tracking with distributed sensors: The focus of attention problem. *Computer Vision and Image Understanding Journal*, (1-2):225–247, 2005. Special Issue on Attention and Performance in Computer Vision.

[22] D. Karapetyan. `http://www.cs.rhul.ac.uk/Research/ToC/publications/Karapetyan/`.

[23] D. Karapetyan, G. Gutin, and B. Goldengorin. Empirical evaluation of construction heuristics for the multidimensional assignment problem. In J. Chan, J. W. Daykin, and M. S. Rahman, editors, *London Algorithmics 2008: Theory and Practice*, Texts in Algorithmics, pages 107–122. College Publications, 2009.

[24] P. Krokhmal, D. Grundel, and P. Pardalos. Asymptotic behavior of the expected optimal value of the multidimensional assignment problem. *Mathematical Programming*, 109(2-3):525–551, 2007.

[25] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 2:83–97, 1955.

[26] Y. Kuroki and T. Matsui. An approximation algorithm for multidimensional assignment problems minimizing the sum of squared errors. *Discrete Applied Mathematics*, 157(9):2124–2135, 2007.

[27] R. Murphey, P. Pardalos, and L. Pitsoulis. A grasp for the multitarget multisensor tracking problem. *Networks, Discrete Mathematics and Theoretical Computer Science Series*, 40:277–302, 1998.

[28] Carlos A. S. Oliveira and Panos M. Pardalos. Randomized parallel algorithms for the multidimensional assignment problem. *Appl. Numer. Math.*, 49:117–133, 2004.

[29] P. M. Pardalos and L. S. Pitsoulis. *Nonlinear assignment problems*. Springer, 2000.

[30] P. M. Pardalos and L. S. Pitsoulis. *Nonlinear Optimization and Applications 2*, chapter Quadratic and Multidimensional Assignment Problems, pages 235–276. Kluwer Academic Publishers, 2000.

[31] E. L. Pasiliao, P. M. Pardalos, and L. S. Pitsoulis. Branch and bound algorithms for the multidimensional assignment problem. *Optimization Methods and Software*, 20(1):127–143, 2005.

[32] W. P. Pierskalla. The multidimensional assignment problem. *Operations Research*, 16:422–431, 1968.

[33] J. Pusztaszeri, P. Rensing, and Th. M. Liebling. Tracking elementary particles near their primary vertex: a combinatorial approach. *Journal of Global Optimization*, 9:41–64, 1996.

[34] R. L. Rardin and R. Uzsoy. Experimental evaluation of heuristic optimization algorithms: A tutorial. *Journal of Heuristics*, 7(3):261–304, 2001.

[35] A. J. Robertson. A set of greedy randomized adaptive local search procedure (grasp) implementations for the multidimensional assignment problem. *Comput. Optim. Appl.*, 19(2):145–164, 2001.

[36] F. Spieksma and G. Woeginger. Geometric three-dimensional assignment problems. *European Journal of Operational Research*, 91:611–618, 1996.

[37] F. C. R. Spieksma. *Nonlinear Assignment Problems, Algorithms and Application*, chapter Multi Index Assignment Problems: Complexity, Approximation, Applications, pages 1–12. Kluwer, 2000.

[38] E-G. Talbi. *Metaheuristics: From Disign to Implementation.* John Wiley & Sons, 2009.

[39] C. J. Veenman, M. J. T. Reinders, and E. Backer. Establishing motion correspondence using extended temporal scope. *Artificial Intelligence*, 145(1-2):227–243, 2003.

Tab. 1: Construction heuristics comparison.

		Solution error, %				Running times, ms			
Inst.	Best	Trivial	Greedy	Max-Regret	ROM	Trivial	Greedy	Max-Regret	ROM
3gp100	504.4	157	$\underline{6}$	6	10	$\underline{0}$	40	799	9
3r150	150.0	4 997	54	$\underline{29}$	34	$\underline{0}$	14	4 253	26
4gp30	145.2	158	9	9	$\underline{2}$	$\underline{0}$	35	206	7
4r80	80.0	4 985	74	$\underline{49}$	76	$\underline{0}$	12	27 285	278
5gp12	66.2	147	13	$\underline{9}$	9	$\underline{0}$	6	36	2
5r40	40.0	4 911	159	$\underline{116}$	169	$\underline{0}$	6	37 214	686
6gp8	41.8	143	25	$\underline{1}$	14	$\underline{0}$	5	33	2
6r22	22.0	5 180	295	$\underline{218}$	310	$\underline{0}$	6	24 750	861
7gp5	25.6	157	27	$\underline{6}$	20	$\underline{0}$	1	8	1
7r14	14.0	5 116	$\underline{377}$	454	396	$\underline{0}$	2	17 032	805
8gp4	19.2	113	21	$\underline{7}$	28	$\underline{0}$	1	8	1
8r9	9.0	5 262	579	$\underline{514}$	543	$\underline{0}$	2	5 604	342
All avg.		2 610	137	$\underline{118}$	134	$\underline{0}$	11	9 769	252
GP avg.		146	17	$\underline{6}$	14	$\underline{0}$	15	182	4
Rand. avg.		5 075	256	$\underline{230}$	255	$\underline{0}$	7	19 356	500
3-AP avg.		2 577	30	$\underline{17}$	22	$\underline{0}$	27	2 526	17
4-AP avg.		2 571	41	$\underline{29}$	39	$\underline{0}$	23	13 745	142
5-AP avg.		2 529	86	$\underline{62}$	89	$\underline{0}$	6	18 625	344
6-AP avg.		2 662	160	$\underline{110}$	162	$\underline{0}$	5	12 391	432
7-AP avg.		2 637	$\underline{202}$	230	208	$\underline{0}$	2	8 520	403
8-AP avg.		2 687	300	$\underline{261}$	286	$\underline{0}$	1	2 806	171
3cq150	1738.5	1 219	41	$\underline{20}$	37	$\underline{0}$	56	4 388	27
3g150	1552.0	865	19	27	$\underline{3}$	$\underline{0}$	53	4 226	28
3p150	14437.2	76	215	122	$\underline{7}$	$\underline{0}$	580	4 318	37
3sr150	1077.8	1 250	42	$\underline{21}$	43	$\underline{0}$	60	4 363	29
4cq50	3034.8	400	27	$\underline{22}$	32	$\underline{0}$	156	3 713	161
4g50	1705.2	492	21	29	$\underline{2}$	$\underline{0}$	217	3 828	148
4p50	20096.8	103	484	278	$\underline{8}$	$\underline{0}$	1 030	3 725	151
4sr50	1496.6	367	25	$\underline{20}$	32	$\underline{0}$	193	3 847	150
5cq30	4727.1	218	20	$\underline{17}$	24	$\underline{0}$	640	9 636	583
5g30	2321.8	340	26	33	$\underline{3}$	$\underline{0}$	936	9 650	604
5p30	55628.5	137	1 017	646	$\underline{8}$	$\underline{0}$	2 711	9 536	619
5sr30	1842.0	196	16	$\underline{13}$	28	$\underline{0}$	666	9 627	615
6cq18	5765.5	142	15	$\underline{15}$	18	$\underline{0}$	426	6 758	267
6g18	2536.0	260	26	27	$\underline{3}$	$\underline{0}$	563	6 802	262
6p18	135515.3	163	2 118	1 263	$\underline{8}$	$\underline{0}$	1 098	6 758	323
6sr18	1856.3	121	13	$\underline{13}$	19	$\underline{0}$	420	6 775	261
7cq12	6663.7	91	14	$\underline{11}$	15	$\underline{0}$	1 037	6 653	924
7g12	3267.2	156	19	23	$\underline{2}$	$\underline{0}$	1 217	6 614	944
7p12	558611.7	346	3 162	1 994	$\underline{9}$	$\underline{0}$	1 872	6 463	335
7sr12	1795.7	78	9	$\underline{9}$	15	$\underline{0}$	980	6 510	268
8cq8	7004.9	62	10	10	$\underline{10}$	$\underline{0}$	465	2 416	130
8g8	3679.5	105	15	21	$\underline{1}$	$\underline{0}$	569	2 446	120
8p8	2233760.0	177	3 605	2 309	$\underline{9}$	$\underline{0}$	710	2 413	140
8sr8	1622.1	52	$\underline{7}$	7	10	$\underline{0}$	474	2 448	132
All avg.		309	457	290	$\underline{14}$	$\underline{0}$	714	5 580	302
Clique avg.		355	21	$\underline{16}$	23	$\underline{0}$	463	5 594	349
Geom. avg.		370	21	27	$\underline{2}$	$\underline{0}$	593	5 594	351
Product avg.		167	1 767	1 102	$\underline{8}$	$\underline{0}$	1 334	5 536	268
SR avg.		344	19	$\underline{14}$	24	$\underline{0}$	465	5 595	242
3-AP avg.		853	79	47	$\underline{22}$	$\underline{0}$	187	4 324	30
4-AP avg.		340	139	87	$\underline{19}$	$\underline{0}$	399	3 778	152
5-AP avg.		223	270	177	$\underline{15}$	$\underline{0}$	1 238	9 612	605
6-AP avg.		171	543	329	$\underline{12}$	$\underline{0}$	627	6 773	278
7-AP avg.		168	801	509	$\underline{10}$	$\underline{0}$	1 276	6 560	618
8-AP avg.		99	909	587	$\underline{8}$	$\underline{0}$	555	2 431	131

Tab. 2: Local search heuristics started from Trivial.

		Solution error, %									
Inst.	Best	2-opt	3-opt	v-opt	1DV	2DV	sDV	1DV$_2$	2DV$_2$	sDV$_3$	sDV$_v$
3gp100	504.4	19.6	10.0	19.8	4.9	4.9	4.9	4.9	4.9	4.6	4.9
3r150	150.0	134.5	16.0	1.5	2.4	2.4	2.4	2.4	2.4	2.1	0.7
4gp30	145.2	17.4	4.2	13.4	11.1	7.9	7.9	10.7	7.9	4.2	7.5
4r80	80.0	115.0	7.3	2.0	20.5	11.5	11.5	18.9	11.5	4.1	1.6
5gp12	66.2	10.6	2.1	8.5	12.5	6.9	6.9	10.1	6.9	1.8	6.9
5r40	40.0	104.5	4.3	3.8	63.0	34.3	34.3	47.3	34.3	3.5	5.3
6gp8	41.8	6.7	2.4	5.3	12.4	5.7	5.0	6.5	5.5	2.4	4.8
6r22	22.0	105.5	0.9	8.6	125.0	62.3	54.5	80.9	55.5	1.8	9.1
7gp5	25.6	6.3	3.9	10.2	21.5	9.0	5.9	5.9	5.1	3.9	5.5
7r14	14.0	95.7	0.0	36.4	244.3	111.4	72.1	92.1	70.0	0.7	16.4
8gp4	19.2	6.8	5.2	10.9	17.2	9.4	6.2	7.8	6.8	5.2	6.2
8r9	9.0	81.1	0.0	67.8	323.3	173.3	60.0	73.3	77.8	0.0	40.0
All avg.		58.6	4.7	15.7	71.5	36.6	22.6	30.1	24.0	2.9	9.1
GP avg.		11.2	4.6	11.3	13.3	7.3	6.1	7.6	6.2	3.7	6.0
Rand. avg.		106.1	4.7	20.0	129.8	65.9	39.1	52.5	41.9	2.0	12.2
3-AP avg.		77.1	13.0	10.6	3.6	3.6	3.6	3.6	3.6	3.3	2.8
4-AP avg.		66.2	5.7	7.7	15.8	9.7	9.7	14.8	9.7	4.2	4.6
5-AP avg.		57.5	3.2	6.1	37.8	20.6	20.6	28.7	20.6	2.7	6.1
6-AP avg.		56.1	1.7	6.9	68.7	34.0	29.8	43.7	30.5	2.1	6.9
7-AP avg.		51.0	2.0	23.3	132.9	60.2	39.0	49.0	37.5	2.3	10.9
8-AP avg.		43.9	2.6	39.4	170.3	91.4	33.1	40.6	42.3	2.6	23.1
3cq150	1738.5	125.1	49.9	22.8	20.1	20.1	20.1	20.1	20.1	19.9	18.9
3g150	1552.0	0.0	0.0	5.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3p150	14437.2	0.1	0.0	15.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3sr150	1077.8	144.2	64.0	28.0	22.0	22.0	22.0	22.0	22.0	21.8	21.3
4cq50	3034.8	52.5	31.3	30.3	23.3	23.1	23.1	23.2	23.1	21.4	20.1
4g50	1705.2	0.0	0.0	11.1	0.2	0.0	0.0	0.0	0.0	0.0	0.0
4p50	20096.8	0.0	0.0	49.6	0.1	0.0	0.0	0.1	0.0	0.0	0.0
4sr50	1496.6	56.8	30.6	31.9	27.2	24.8	24.8	27.2	24.8	23.4	23.9
5cq30	4727.1	30.9	18.7	21.4	16.9	16.6	16.6	16.8	16.6	15.5	16.1
5g30	2321.8	0.0	0.0	9.2	0.2	0.0	0.0	0.0	0.0	0.0	0.0
5p30	55628.5	0.0	0.0	53.2	0.1	0.0	0.0	0.0	0.0	0.0	0.0
5sr30	1842.0	38.3	19.0	23.9	21.7	20.4	20.4	21.1	20.4	17.6	18.3
6cq18	5765.5	17.6	12.2	16.1	11.5	10.3	11.6	11.3	10.3	10.1	11.1
6g18	2536.0	0.0	0.0	15.4	0.5	0.0	0.0	0.0	0.0	0.0	0.0
6p18	135515.3	0.0	0.0	98.3	0.2	0.0	0.0	0.0	0.0	0.0	0.0
6sr18	1856.3	20.9	11.9	17.4	12.7	13.9	13.6	12.7	13.9	11.5	12.6
7cq12	6663.7	11.9	5.3	10.4	8.0	7.0	5.9	7.1	6.9	5.7	5.8
7g12	3267.2	0.0	0.0	9.9	0.1	0.0	0.0	0.0	0.0	0.0	0.0
7p12	558611.7	0.0	0.0	123.6	0.2	0.0	0.0	0.0	0.0	0.0	0.0
7sr12	1795.7	12.1	7.6	11.0	8.5	10.1	7.1	8.3	10.1	5.9	7.0
8cq8	7004.9	6.4	3.0	8.5	6.4	4.4	4.8	5.3	4.1	2.2	4.7
8g8	3679.5	0.0	0.0	9.1	0.2	0.0	0.0	0.0	0.0	0.0	0.0
8p8	2233760.0	0.0	0.0	143.8	0.1	0.0	0.0	0.0	0.0	0.0	0.0
8sr8	1622.1	6.6	2.6	7.4	5.7	5.0	4.7	4.9	4.4	3.5	4.7
All avg.		21.8	10.7	32.2	7.8	7.4	7.3	7.5	7.4	6.6	6.9
Clique avg.		40.7	20.0	18.2	14.4	13.6	13.7	14.0	13.5	12.5	12.8
Geom. avg.		0.0	0.0	10.1	0.2	0.0	0.0	0.0	0.0	0.0	0.0
Product avg.		0.0	0.0	80.6	0.1	0.0	0.0	0.0	0.0	0.0	0.0
SR avg.		46.5	22.6	19.9	16.3	16.1	15.4	16.0	16.0	13.9	14.7
3-AP avg.		67.3	28.5	17.9	10.5	10.5	10.5	10.5	10.5	10.4	10.1
4-AP avg.		27.3	15.5	30.7	12.7	12.0	12.0	12.6	12.0	11.2	11.0
5-AP avg.		17.3	9.4	26.9	9.7	9.3	9.3	9.5	9.3	8.3	8.6
6-AP avg.		9.6	6.0	36.8	6.2	6.1	6.3	6.0	6.1	5.4	6.0
7-AP avg.		6.0	3.2	38.7	4.2	4.3	3.2	3.8	4.3	2.9	3.2
8-AP avg.		3.2	1.4	42.2	3.1	2.4	2.4	2.6	2.1	1.4	2.4

Tab. 3: Local search heuristics started from Trivial.

	Running time, ms									
Inst.	2-opt	3-opt	v-opt	1DV	2DV	*s*DV	$1DV_2$	$2DV_2$	sDV_3	sDV_v
3gp100	6.2	820.6	181.8	14.3	14.0	16.5	18.4	16.7	430.6	79.0
3r150	19.8	1 737.9	65.7	17.6	18.7	17.1	22.7	18.9	147.8	45.4
4gp30	1.5	150.3	45.0	0.7	1.2	1.1	1.4	1.4	116.9	17.5
4r80	10.5	987.5	64.5	7.9	18.0	15.3	11.2	18.4	344.8	98.2
5gp12	0.3	38.5	3.6	0.2	0.4	0.5	0.5	0.5	30.6	1.6
5r40	16.9	425.9	34.3	2.3	7.2	6.3	4.6	8.6	386.9	35.3
6gp8	0.2	57.2	2.5	0.2	0.3	0.4	0.4	0.5	42.0	1.3
6r22	2.2	218.9	16.7	0.9	2.6	3.9	1.9	4.3	259.0	22.7
7gp5	0.1	48.9	0.9	0.1	0.2	0.3	0.1	0.3	40.0	0.9
7r14	1.4	237.1	12.0	0.4	1.6	2.9	1.8	3.0	210.9	15.5
8gp4	0.1	117.5	0.8	0.2	0.3	0.6	0.2	0.3	72.3	0.9
8r9	0.9	191.9	6.7	0.3	1.1	2.3	1.0	3.1	177.7	7.1
All avg.	5.0	419.4	36.2	3.8	5.5	5.6	5.3	6.3	188.3	27.1
GP avg.	1.4	205.5	39.1	2.6	2.7	3.2	3.5	3.3	122.1	16.9
Rand. avg.	8.6	633.2	33.3	4.9	8.2	7.9	7.2	9.4	254.5	37.4
3-AP avg.	13.0	1 279.2	123.8	16.0	16.4	16.8	20.5	17.8	289.2	62.2
4-AP avg.	6.0	568.9	54.7	4.3	9.6	8.2	6.3	9.9	230.8	57.8
5-AP avg.	8.6	232.2	19.0	1.3	3.8	3.4	2.5	4.5	208.7	18.5
6-AP avg.	1.2	138.1	9.6	0.5	1.5	2.1	1.1	2.4	150.5	12.0
7-AP avg.	0.7	143.0	6.5	0.3	0.9	1.6	1.0	1.6	125.5	8.2
8-AP avg.	0.5	154.7	3.8	0.3	0.7	1.4	0.6	1.7	125.0	4.0
3cq150	22.1	4 366.5	1 388.4	42.1	39.3	34.9	41.0	46.0	1 503.9	497.6
3g150	19.0	2 229.3	780.0	26.2	28.1	25.5	37.2	33.0	1 299.5	201.2
3p150	15.4	2 149.7	847.1	82.0	89.8	89.7	96.0	101.9	1 730.1	458.6
3sr150	21.7	3 949.9	1 157.5	36.0	37.5	37.9	41.2	47.1	1 400.9	469.6
4cq50	6.1	872.0	308.9	3.8	8.5	7.3	6.1	10.8	468.0	167.2
4g50	5.3	542.9	251.2	3.7	5.9	5.9	6.7	6.6	273.0	87.3
4p50	5.7	586.6	251.2	7.3	14.2	13.6	13.4	15.7	441.5	95.5
4sr50	5.6	1 009.3	296.4	3.3	7.4	6.2	6.0	7.9	424.3	111.6
5cq30	4.6	1 087.3	177.7	2.0	5.2	5.5	3.3	6.0	560.0	63.5
5g30	3.7	673.9	182.5	1.8	4.1	4.0	3.6	5.7	319.8	41.8
5p30	4.5	762.8	103.6	2.7	10.1	9.5	6.1	12.2	580.3	44.1
5sr30	4.8	1 115.4	163.5	1.9	4.7	4.5	3.6	6.3	667.7	63.2
6cq18	3.5	1 205.9	63.4	1.0	2.7	3.7	1.5	3.1	630.2	26.6
6g18	2.0	731.6	55.2	0.9	1.8	2.7	1.9	2.4	346.3	18.1
6p18	3.1	929.8	31.1	1.3	3.8	5.4	2.5	5.2	658.3	19.9
6sr18	2.3	1 369.7	59.9	0.9	2.9	3.0	1.5	3.4	778.4	34.4
7cq12	1.7	1 658.3	31.7	0.6	2.0	3.4	1.2	2.9	728.5	12.6
7g12	1.4	1 048.3	28.2	0.6	1.3	2.4	1.1	2.0	555.4	11.1
7p12	2.1	1 324.4	17.5	0.8	2.4	6.4	1.8	3.9	1 088.9	14.6
7sr12	1.9	1 622.4	40.9	0.7	2.0	3.5	1.1	2.5	965.6	11.0
8cq8	1.1	2 112.3	13.3	0.5	1.5	2.8	1.0	2.0	1 909.5	8.5
8g8	1.0	1 675.5	15.6	0.4	0.8	2.1	0.8	1.2	728.5	7.2
8p8	1.7	2 051.4	7.6	0.4	1.2	3.1	0.9	1.8	1 492.9	7.9
8sr8	1.3	2 439.9	16.4	0.3	1.3	2.9	1.0	1.8	1 252.7	8.1
All avg.	5.9	1 563.1	262.0	9.2	11.6	11.9	11.7	13.8	866.8	103.4
Clique avg.	6.5	1 883.7	330.6	8.3	9.9	9.6	9.0	11.8	966.7	129.4
Geom. avg.	5.4	1 150.2	218.8	5.6	7.0	7.1	8.5	8.5	587.1	61.1
Product avg.	5.4	1 300.8	209.7	15.8	20.2	21.3	20.1	23.4	998.7	106.8
SR avg.	6.3	1 917.8	289.1	7.2	9.3	9.7	9.1	11.5	914.9	116.3
3-AP avg.	19.5	3 173.8	1 043.3	46.6	48.7	47.0	53.8	57.0	1 483.6	406.8
4-AP avg.	5.7	752.7	276.9	4.5	9.0	8.2	8.0	10.2	401.7	115.4
5-AP avg.	4.4	909.9	156.8	2.1	6.0	5.9	4.2	7.5	532.0	53.2
6-AP avg.	2.7	1 059.2	52.4	1.0	2.8	3.7	1.9	3.5	603.3	24.8
7-AP avg.	1.7	1 413.4	29.6	0.7	1.9	3.9	1.3	2.8	834.6	12.3
8-AP avg.	1.2	2 069.7	13.2	0.4	1.2	2.7	0.9	1.7	1 345.9	7.9

Tab. 4: Local search heuristics started from Greedy.

	Solution error, %								Running times, ms							
Inst.	2-opt	1DV	2DV	sDV	1DV$_2$	2DV$_2$	sDV$_3$	sDV$_v$	2-opt	1DV	2DV	sDV	1DV$_2$	2DV$_2$	sDV$_3$	sDV$_v$
3gp100	4.3	3.4	3.4	3.4	3.4	3.4	3.3	3.4	0.04	0.04	0.04	0.04	0.05	0.05	0.36	0.09
3r150	16.7	1.2	1.2	1.2	1.2	1.2	0.8	0.7	0.02	0.02	0.02	0.03	0.02	0.03	0.11	0.05
4gp30	4.5	3.7	3.6	3.6	3.6	3.6	2.6	3.6	0.04	0.03	0.04	0.04	0.04	0.04	0.11	0.05
4r80	15.8	7.9	6.1	6.1	7.9	6.1	2.6	1.5	0.01	0.02	0.02	0.02	0.02	0.02	0.21	0.08
5gp12	5.4	6.3	4.5	4.5	5.3	4.5	1.8	4.5	0.01	0.01	0.01	0.01	0.01	0.01	0.03	0.01
5r40	18.5	19.8	13.5	13.5	15.0	13.5	2.3	3.5	0.01	0.01	0.01	0.01	0.01	0.01	0.18	0.04
6gp8	4.1	8.9	5.5	4.3	6.0	4.5	2.4	3.8	0.01	0.01	0.01	0.01	0.01	0.01	0.04	0.01
6r22	25.9	44.1	28.6	26.4	26.8	27.3	2.7	8.6	0.01	0.01	0.01	0.01	0.01	0.01	0.21	0.02
7gp5	5.5	11.3	7.0	5.9	6.6	5.9	3.5	5.1	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00
7r14	37.9	88.6	55.7	33.6	51.4	44.3	0.0	15.0	0.00	0.00	0.00	0.00	0.00	0.00	0.14	0.01
8gp4	4.2	11.5	5.2	3.6	4.2	3.6	3.1	3.6	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.00
8r9	40.0	158.9	107.8	54.4	65.6	65.6	0.0	30.0	0.00	0.00	0.00	0.00	0.00	0.00	0.13	0.01
All avg.	15.2	30.5	20.2	13.4	16.4	15.3	2.1	6.9	0.01	0.01	0.01	0.01	0.01	0.01	0.14	0.03
GP avg.	4.7	7.5	4.9	4.2	4.8	4.3	2.8	4.0	0.02	0.02	0.02	0.02	0.02	0.02	0.11	0.03
Rand. avg.	25.8	53.4	35.5	22.5	28.0	26.3	1.4	9.9	0.01	0.01	0.01	0.01	0.01	0.01	0.16	0.03
3-AP avg.	10.5	2.3	2.3	2.3	2.3	2.3	2.1	2.0	0.03	0.03	0.03	0.04	0.04	0.04	0.24	0.07
4-AP avg.	10.1	5.8	4.9	4.9	5.7	4.9	2.6	2.5	0.02	0.03	0.03	0.03	0.03	0.03	0.16	0.06
5-AP avg.	12.0	13.0	9.0	9.0	10.1	9.0	2.0	4.0	0.01	0.01	0.01	0.01	0.01	0.01	0.11	0.02
6-AP avg.	15.0	26.5	17.1	15.3	16.4	15.9	2.6	6.2	0.01	0.01	0.01	0.01	0.01	0.01	0.13	0.01
7-AP avg.	21.7	49.9	31.4	19.7	29.0	25.1	1.8	10.0	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.01
8-AP avg.	22.1	85.2	56.5	29.0	34.9	34.6	1.6	16.8	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.01
3cq150	26.8	8.1	8.1	8.1	8.1	8.1	8.0	8.0	0.07	0.07	0.07	0.07	0.08	0.08	1.18	0.26
3g150	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.07	0.07	0.07	0.07	0.08	0.08	1.09	0.22
3p150	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.61	0.65	0.66	0.66	0.66	0.66	1.92	0.96
3sr150	29.9	9.8	9.8	9.8	9.8	9.8	9.4	9.1	0.07	0.07	0.07	0.07	0.08	0.09	1.49	0.26
4cq50	19.0	11.6	11.6	11.6	11.6	11.6	11.3	11.6	0.16	0.16	0.16	0.16	0.16	0.16	0.44	0.21
4g50	0.0	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.22	0.22	0.22	0.22	0.22	0.22	0.43	0.29
4p50	0.1	0.2	0.1	0.1	0.1	0.1	0.0	0.1	1.04	1.04	1.04	1.04	1.04	1.05	1.39	1.12
4sr50	20.0	10.9	11.3	11.3	10.9	11.3	10.3	11.0	0.19	0.19	0.20	0.20	0.20	0.20	0.47	0.25
5cq30	14.2	9.6	9.5	9.5	9.6	9.5	9.3	9.4	0.64	0.64	0.64	0.64	0.64	0.64	1.03	0.68
5g30	0.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.94	0.94	0.94	0.94	0.94	0.94	1.26	0.97
5p30	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	2.72	2.71	2.72	2.72	2.72	2.72	3.23	2.76
5sr30	11.7	8.9	8.5	8.5	8.3	8.5	7.1	8.5	0.67	0.67	0.67	0.67	0.67	0.67	1.23	0.69
6cq18	9.8	8.2	7.8	7.5	7.9	7.8	6.3	7.3	0.43	0.43	0.43	0.43	0.43	0.43	1.08	0.44
6g18	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.56	0.56	0.56	0.57	0.56	0.57	0.90	0.58
6p18	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	1.10	1.10	1.10	1.10	1.10	1.10	1.69	1.12
6sr18	9.7	8.6	8.2	8.2	8.5	8.2	6.5	7.8	0.42	0.42	0.42	0.42	0.42	0.42	1.15	0.44
7cq12	7.1	5.7	5.0	5.1	5.1	5.0	4.0	4.9	1.04	1.04	1.04	1.04	1.04	1.04	2.20	1.05
7g12	0.0	0.5	0.1	0.0	0.0	0.0	0.0	0.0	1.22	1.22	1.22	1.22	1.22	1.22	1.77	1.23
7p12	0.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	1.88	1.87	1.87	1.88	1.87	1.88	2.90	1.89
7sr12	6.5	5.7	5.1	5.2	5.6	5.1	4.0	5.0	0.98	0.98	0.98	0.98	0.98	0.98	2.15	0.99
8cq8	4.7	4.1	3.1	2.8	3.7	2.7	2.2	2.6	0.47	0.47	0.47	0.47	0.47	0.47	1.97	0.47
8g8	0.0	0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.57	0.57	0.57	0.57	0.57	0.57	1.38	0.58
8p8	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.71	0.71	0.71	0.71	0.71	0.71	2.11	0.72
8sr8	3.2	3.7	2.8	2.6	2.6	2.5	2.1	2.4	0.47	0.47	0.47	0.48	0.47	0.48	1.72	0.48
All avg.	6.8	4.1	3.8	3.8	3.8	3.8	3.4	3.7	0.72	0.72	0.72	0.72	0.72	0.72	1.51	0.78
Clique avg.	13.6	7.9	7.5	7.4	7.6	7.5	6.8	7.3	0.47	0.47	0.47	0.47	0.47	0.47	1.32	0.52
Geom. avg.	0.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.60	0.60	0.60	0.60	0.60	0.60	1.14	0.65
Product avg.	0.1	0.2	0.0	0.0	0.0	0.0	0.0	0.0	1.34	1.35	1.35	1.35	1.35	1.35	2.21	1.43
SR avg.	13.5	7.9	7.6	7.6	7.6	7.6	6.6	7.3	0.47	0.47	0.47	0.47	0.47	0.47	1.37	0.52
3-AP avg.	14.2	4.5	4.5	4.5	4.5	4.5	4.4	4.3	0.20	0.22	0.22	0.22	0.22	0.23	1.42	0.43
4-AP avg.	9.8	5.7	5.7	5.7	5.6	5.7	5.4	5.7	0.40	0.40	0.41	0.41	0.40	0.41	0.68	0.47
5-AP avg.	6.5	4.8	4.5	4.5	4.5	4.5	4.1	4.5	1.24	1.24	1.24	1.24	1.24	1.24	1.69	1.27
6-AP avg.	4.9	4.4	4.0	3.9	4.1	4.0	3.2	3.8	0.63	0.63	0.63	0.63	0.63	0.63	1.21	0.64
7-AP avg.	3.4	3.1	2.6	2.6	2.7	2.5	2.0	2.5	1.28	1.28	1.28	1.28	1.28	1.28	2.26	1.29
8-AP avg.	2.0	2.2	1.5	1.4	1.6	1.3	1.1	1.3	0.56	0.55	0.56	0.56	0.56	0.56	1.80	0.56

Tab. 5: Chain metaheuristic started from Trivial, Greedy and ROM. 5 seconds given. 1 — 2-opt, 2 — 1DV, 3 — 2DV, 4 — sDV, 5 — 1DV$_2$, 6 — 2DV$_2$, 7 — sDV$_3$, 8 — sDV$_v$.

	Solution error, %																							
	Trivial								Greedy								ROM							
Inst.	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
3gp100	15.3	1.8	1.8	1.8	1.8	1.8	2.8	2.5	5.3	1.7	1.7	1.7	1.8	1.8	2.9	2.3	9.8	1.9	1.9	1.9	1.9	1.9	2.6	2.3
3r150	77.7	0.0	0.0	0.0	0.0	0.0	0.1	0.0	41.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	33.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4gp30	7.0	1.6	1.1	1.1	1.8	1.1	0.8	1.4	6.3	1.9	0.8	0.8	1.8	0.9	0.8	1.4	2.2	1.7	0.9	0.9	1.7	1.0	0.8	1.4
4r80	55.0	4.4	1.9	1.9	4.1	2.3	0.4	0.0	41.6	4.6	1.6	1.6	4.5	1.8	0.8	0.0	57.0	4.3	2.0	2.0	4.3	2.0	0.9	0.0
5gp12	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
5r40	40.8	18.5	8.0	8.0	16.3	8.0	0.0	0.0	34.0	19.3	8.0	8.0	13.5	8.5	0.0	0.0	40.3	19.3	8.0	8.0	15.8	8.8	0.5	0.0
6gp8	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4
6r22	20.5	30.0	10.9	6.4	15.5	8.2	0.0	0.0	19.1	27.7	11.8	5.5	15.5	9.1	0.0	0.0	15.5	32.7	13.6	8.6	15.0	9.5	0.0	0.0
7gp5	3.9	3.9	3.9	3.9	3.9	3.9	3.9	3.9	3.1	3.5	3.5	3.5	3.5	3.1	3.5	3.5	3.5	3.9	3.9	3.9	3.9	3.9	3.9	3.9
7r14	2.9	33.6	11.4	2.1	6.4	3.6	0.0	0.0	3.6	33.6	10.7	2.1	5.7	2.1	0.0	0.0	4.3	35.7	7.9	0.7	2.1	3.6	0.0	0.0
8gp4	2.1	5.2	4.7	4.2	2.1	3.6	5.2	5.2	0.5	3.1	3.1	3.1	2.6	1.6	3.1	2.6	1.0	4.7	4.7	3.6	2.6	4.2	4.7	4.7
8r9	0.0	25.6	4.4	0.0	0.0	0.0	0.0	0.0	0.0	22.2	2.2	0.0	0.0	0.0	0.0	0.0	0.0	26.7	4.4	0.0	0.0	0.0	0.0	0.0
All avg.	19.1	10.7	4.3	2.8	4.6	3.0	1.4	1.4	13.2	10.1	4.0	2.5	4.4	2.7	1.2	1.1	14.3	11.2	4.3	2.8	4.3	3.2	1.4	1.3
GP avg.	5.4	2.7	2.6	2.5	2.2	2.4	2.8	2.8	3.2	2.4	2.2	2.2	2.3	1.9	2.4	2.3	3.4	2.7	2.5	2.4	2.3	2.5	2.6	2.7
Rand. avg.	32.8	18.7	6.1	3.1	7.0	3.7	0.1	0.0	23.3	17.9	5.7	2.9	6.5	3.6	0.1	0.0	25.1	19.8	6.0	3.2	6.2	4.0	0.2	0.0
3-AP avg.	46.5	0.9	0.9	0.9	0.9	0.9	1.4	1.2	23.3	0.9	0.9	0.9	0.9	0.9	1.4	1.1	21.7	0.9	0.9	0.9	1.0	1.0	1.3	1.1
4-AP avg.	31.0	3.0	1.5	1.5	3.0	1.7	0.6	0.7	23.9	3.3	1.2	1.2	3.1	1.3	0.8	0.7	29.6	3.0	1.4	1.4	3.0	1.5	0.8	0.7
5-AP avg.	21.1	10.0	4.8	4.8	8.9	4.8	0.8	0.8	17.8	10.4	4.8	4.8	7.5	5.0	0.8	0.8	20.9	10.4	4.8	4.8	8.6	5.1	1.0	0.8
6-AP avg.	11.4	16.2	6.7	4.4	8.9	5.3	1.2	1.2	10.7	15.1	7.1	3.9	8.9	5.7	1.2	1.2	8.9	17.6	8.0	5.5	8.7	6.0	1.2	1.2
7-AP avg.	3.4	18.7	7.7	3.0	5.2	3.7	2.0	2.0	3.3	18.5	7.1	2.8	4.6	2.6	1.8	1.8	3.9	19.8	5.9	2.3	3.0	3.7	2.0	2.0
8-AP avg.	1.0	15.4	4.6	2.1	1.0	1.8	2.6	2.6	0.3	12.7	2.7	1.6	1.3	0.8	1.6	1.3	0.5	15.7	4.6	1.8	1.3	2.1	2.3	2.3
3cq150	80.7	6.2	6.2	6.2	6.7	6.7	17.0	9.8	38.2	6.0	6.0	6.0	6.1	6.0	8.4	6.3	36.8	6.4	6.4	6.4	6.5	6.5	15.8	11.3
3g150	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3p150	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3sr150	96.0	7.0	7.0	7.1	7.6	7.6	18.3	11.8	41.0	7.4	7.4	7.4	7.9	7.9	9.1	7.4	42.8	6.7	6.7	6.8	7.0	7.2	17.8	11.4
4cq50	27.7	5.4	5.8	5.8	5.6	6.1	12.7	9.5	22.5	5.4	5.7	5.7	6.1	5.8	9.8	7.4	26.4	5.1	5.2	5.0	5.4	5.6	13.0	8.0
4g50	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4p50	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4sr50	31.9	6.4	7.1	7.1	7.3	7.4	14.4	8.8	23.3	6.6	7.2	7.2	7.6	7.4	9.2	7.6	30.0	6.5	7.1	7.1	7.3	7.3	13.5	10.4
5cq30	11.6	2.7	2.5	2.4	2.7	2.5	8.3	4.4	11.8	2.3	2.7	2.6	2.9	2.8	5.6	3.9	11.9	2.6	2.8	2.6	2.9	3.1	9.0	4.8
5g30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5p30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5sr30	15.3	4.1	4.1	3.8	5.1	4.2	10.5	6.6	13.5	4.2	4.0	4.0	4.9	4.2	7.4	6.0	14.9	4.3	4.7	4.5	4.6	4.7	9.8	5.9
6cq18	3.2	0.3	0.2	0.4	0.5	0.3	5.9	1.4	3.3	0.3	0.3	0.4	0.4	0.5	4.4	1.5	2.7	0.4	0.3	0.4	0.6	0.2	6.5	1.3
6g18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6p18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6sr18	4.1	0.7	0.5	0.8	1.2	1.1	7.6	1.9	4.0	1.0	0.9	0.7	0.9	0.9	5.7	2.5	4.2	1.1	0.7	1.0	1.2	0.7	7.1	2.4
7cq12	0.5	0.0	0.0	0.0	0.0	0.0	3.8	0.3	0.4	0.0	0.0	0.0	0.0	0.0	2.7	0.4	0.4	0.0	0.0	0.1	0.0	0.0	4.3	0.2
7g12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7p12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7sr12	0.6	0.0	0.0	0.0	0.1	0.0	4.6	0.4	0.7	0.0	0.0	0.1	0.0	0.1	3.4	0.6	0.4	0.0	0.0	0.1	0.0	0.1	5.1	0.3
8cq8	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.9	0.0
8g8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8p8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8sr8	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.4	0.0
All avg.	11.3	1.4	1.4	1.4	1.5	1.5	4.5	2.3	6.6	1.4	1.4	1.4	1.5	1.5	2.9	1.8	7.1	1.4	1.4	1.4	1.5	1.5	4.4	2.3
Clique avg.	20.6	2.4	2.4	2.5	2.6	2.6	8.3	4.2	12.7	2.3	2.4	2.5	2.6	2.5	5.6	3.2	13.0	2.4	2.4	2.4	2.6	2.6	8.4	4.3
Geom. avg.	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Product avg.	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
SR avg.	24.6	3.0	3.1	3.1	3.5	3.4	9.6	4.9	13.7	3.2	3.2	3.2	3.5	3.4	6.1	4.0	15.4	3.1	3.2	3.2	3.4	3.3	9.3	5.0
3-AP avg.	44.2	3.3	3.3	3.3	3.6	3.6	8.8	5.4	19.8	3.4	3.4	3.4	3.5	3.5	4.4	3.4	19.9	3.3	3.3	3.3	3.4	3.4	8.4	5.7
4-AP avg.	14.9	3.0	3.2	3.2	3.2	3.4	6.8	4.6	11.4	3.0	3.2	3.2	3.4	3.3	4.8	3.8	14.1	2.9	3.1	3.0	3.2	3.2	6.6	4.6
5-AP avg.	6.7	1.7	1.7	1.6	1.9	1.7	4.7	2.7	6.3	1.6	1.7	1.6	2.0	1.8	3.2	2.5	6.7	1.8	1.9	1.8	1.9	1.9	4.7	2.7
6-AP avg.	1.8	0.3	0.2	0.3	0.4	0.3	3.4	0.8	1.8	0.3	0.3	0.3	0.3	0.3	2.5	1.0	1.7	0.4	0.3	0.4	0.5	0.2	3.4	0.9
7-AP avg.	0.3	0.0	0.0	0.0	0.0	0.0	2.1	0.2	0.3	0.0	0.0	0.0	0.0	0.0	1.5	0.2	0.2	0.0	0.0	0.0	0.0	0.0	2.4	0.1
8-AP avg.	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.1	0.0

Tab. 6: Chain metaheuristic started from Trivial, Greedy and ROM. 10 seconds given. 1 — 2-opt, 2 — 1DV, 3 — 2DV, 4 — sDV, 5 — 1DV_2, 6 — 2DV_2, 7 — $s\text{DV}_3$, 8 — $s\text{DV}_\text{v}$.

	Solution error, %																							
	Trivial								Greedy								ROM							
Inst.	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
3gp100	15.1	1.6	1.6	1.6	1.7	1.7	2.3	2.2	5.3	1.6	1.6	1.6	1.6	1.6	2.5	2.1	9.8	1.6	1.6	1.6	1.8	1.7	2.2	2.1
3r150	75.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	41.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	33.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4gp30	6.5	1.4	0.8	0.8	1.3	1.0	0.7	1.3	6.2	1.7	0.8	0.8	1.5	0.8	0.7	1.1	2.2	1.4	0.8	0.8	1.4	0.8	0.7	1.2
4r80	52.1	3.9	1.1	1.0	3.6	1.1	0.1	0.0	41.4	3.9	1.0	1.0	4.3	1.1	0.4	0.0	55.0	4.0	1.1	1.1	3.4	1.3	0.4	0.0
5gp12	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
5r40	36.5	16.3	6.5	5.8	13.0	6.8	0.0	0.0	32.3	18.8	7.0	7.0	13.0	7.0	0.0	0.0	36.8	16.5	6.8	6.8	13.8	7.3	0.0	0.0
6gp8	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4
6r22	16.8	27.7	9.1	5.0	12.3	7.7	0.0	0.0	15.5	26.8	11.4	4.5	13.2	8.2	0.0	0.0	14.1	30.0	10.5	5.9	12.3	8.6	0.0	0.0
7gp5	3.5	3.9	3.9	3.9	3.9	3.9	3.9	3.9	3.1	3.5	3.5	3.5	3.5	3.1	3.5	3.5	2.7	3.9	3.9	3.9	3.9	3.5	3.9	3.9
7r14	1.4	29.3	7.1	1.4	2.9	2.9	0.0	0.0	0.7	31.4	6.4	0.7	4.3	0.0	0.0	0.0	2.9	29.3	5.7	0.7	0.0	2.1	0.0	0.0
8gp4	1.6	5.2	4.7	3.6	1.0	2.1	5.2	3.6	0.5	3.1	2.1	2.6	1.6	1.6	2.6	2.6	1.0	4.7	4.7	3.6	1.0	2.1	4.2	4.2
8r9	0.0	23.3	1.1	0.0	0.0	0.0	0.0	0.0	0.0	15.6	1.1	0.0	0.0	0.0	0.0	0.0	0.0	22.2	4.4	0.0	0.0	0.0	0.0	0.0
All avg.	17.7	9.7	3.3	2.3	3.6	2.6	1.3	1.2	12.5	9.2	3.2	2.1	3.9	2.3	1.1	1.1	13.5	9.8	3.6	2.4	3.5	2.6	1.3	1.3
GP avg.	5.1	2.7	2.5	2.3	2.0	2.1	2.7	2.5	3.2	2.3	2.0	2.1	2.0	1.8	2.2	2.2	3.3	2.6	2.5	2.3	2.0	2.0	2.5	2.5
Rand. avg.	30.4	16.7	4.2	2.2	5.3	3.1	0.0	0.0	21.9	16.1	4.5	2.2	5.8	2.7	0.1	0.0	23.7	17.0	4.7	2.4	4.9	3.2	0.1	0.0
3-AP avg.	45.2	0.8	0.8	0.8	0.9	0.9	1.1	1.1	23.3	0.8	0.8	0.8	0.8	0.8	1.3	1.0	21.7	0.8	0.8	0.8	0.9	0.9	1.1	1.1
4-AP avg.	29.3	2.7	1.0	0.9	2.5	1.0	0.4	0.7	23.8	2.8	0.9	0.9	2.9	0.9	0.5	0.6	28.6	2.7	1.0	1.0	2.4	1.0	0.5	0.6
5-AP avg.	19.0	8.9	4.0	3.6	7.3	4.1	0.8	0.8	16.9	10.1	4.3	4.3	7.3	4.3	0.8	0.8	19.1	9.0	4.1	4.1	7.6	4.4	0.8	0.8
6-AP avg.	9.6	15.1	5.7	3.7	7.3	5.1	1.2	1.2	8.9	14.6	6.9	3.5	7.8	5.3	1.2	1.2	8.2	16.2	6.4	4.2	7.3	5.5	1.2	1.2
7-AP avg.	2.5	16.6	5.5	2.7	3.4	3.4	2.0	2.0	1.9	17.5	5.0	2.1	3.9	1.6	1.8	1.8	2.8	16.6	4.8	2.3	2.0	2.8	2.0	2.0
8-AP avg.	0.8	14.3	2.9	1.8	0.5	1.0	2.6	1.8	0.3	9.3	1.6	1.3	0.8	0.8	1.3	1.3	0.5	13.5	4.6	1.8	0.5	1.0	2.1	2.1
3cq150	79.8	5.4	5.4	5.4	5.9	5.9	13.3	8.3	38.2	5.6	5.6	5.6	5.7	5.8	7.9	6.1	36.8	6.1	5.9	5.9	6.3	6.2	12.7	8.2
3g150	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3p150	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3sr150	93.9	6.3	6.3	6.3	6.7	6.7	15.8	10.2	41.0	6.2	6.2	6.2	6.6	6.6	8.3	7.2	42.8	6.6	6.5	6.5	6.7	6.7	14.5	9.1
4cq50	26.2	5.0	5.0	4.9	5.2	5.3	9.9	6.5	22.4	4.9	5.3	5.2	5.4	5.5	8.9	7.0	25.5	4.6	4.8	4.8	5.2	4.9	11.3	7.1
4g50	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4p50	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4sr50	30.8	5.8	6.6	6.6	6.5	6.9	11.2	8.2	23.3	6.4	6.5	6.5	6.7	6.6	8.9	6.7	29.3	6.2	6.1	6.1	6.9	6.7	11.3	9.2
5cq30	10.9	2.2	1.9	2.0	2.0	2.1	6.9	4.2	11.0	1.9	2.2	2.2	2.3	2.5	5.1	3.4	11.4	2.4	2.3	2.3	2.4	2.4	7.3	3.7
5g30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5p30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5sr30	13.8	3.7	3.5	3.2	3.9	3.5	8.9	5.0	12.2	3.9	3.5	3.5	4.0	3.7	6.3	4.9	14.0	4.0	4.0	4.0	3.8	4.2	8.6	4.8
6cq18	2.5	0.2	0.1	0.3	0.4	0.2	4.1	0.8	2.7	0.3	0.2	0.0	0.2	0.4	3.5	0.8	2.3	0.2	0.1	0.2	0.2	0.2	4.8	1.0
6g18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6p18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6sr18	3.4	0.4	0.4	0.4	0.7	0.8	5.1	1.0	3.4	0.7	0.7	0.3	0.6	0.6	4.8	2.0	3.8	0.5	0.6	0.5	0.8	0.6	5.5	1.8
7cq12	0.2	0.0	0.0	0.0	0.0	0.0	2.7	0.1	0.2	0.0	0.0	0.0	0.0	0.0	2.1	0.1	0.2	0.0	0.0	0.0	0.0	0.0	3.4	0.1
7g12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7p12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7sr12	0.2	0.0	0.0	0.0	0.0	0.0	3.5	0.1	0.5	0.0	0.0	0.1	0.0	0.0	2.4	0.2	0.3	0.0	0.0	0.1	0.0	0.0	4.1	0.2
8cq8	0.0	0.0	0.0	0.0	0.0	0.0	1.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.4	0.0
8g8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8p8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8sr8	0.0	0.0	0.0	0.0	0.0	0.0	1.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.1	0.0
All avg.	10.9	1.2	1.2	1.2	1.3	1.3	3.5	1.9	6.5	1.3	1.3	1.2	1.3	1.3	2.5	1.6	6.9	1.3	1.3	1.3	1.4	1.3	3.6	1.9
Clique avg.	19.9	2.1	2.1	2.1	2.2	2.3	6.4	3.3	12.4	2.1	2.2	2.2	2.3	2.4	4.8	2.9	12.7	2.2	2.2	2.2	2.4	2.3	6.8	3.3
Geom. avg.	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Product avg.	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
SR avg.	23.7	2.7	2.8	2.7	3.0	3.0	7.7	4.1	13.4	2.9	2.8	2.8	3.0	2.9	5.3	3.5	15.0	2.9	2.9	2.9	3.0	3.0	7.7	4.2
3-AP avg.	43.4	2.9	2.9	2.9	3.1	3.1	7.3	4.6	19.8	3.0	3.0	3.0	3.1	3.1	4.0	3.3	19.9	3.2	3.1	3.1	3.2	3.2	6.8	4.3
4-AP avg.	14.2	2.7	2.9	2.9	2.9	3.1	5.3	3.7	11.4	2.8	2.9	2.9	3.0	3.0	4.4	3.4	13.7	2.7	2.7	2.7	3.0	2.9	5.7	4.1
5-AP avg.	6.2	1.5	1.3	1.3	1.5	1.4	3.9	2.3	5.8	1.5	1.4	1.4	1.6	1.6	2.8	2.1	6.4	1.6	1.6	1.6	1.6	1.7	4.0	2.1
6-AP avg.	1.5	0.1	0.1	0.2	0.3	0.3	2.3	0.5	1.5	0.2	0.2	0.1	0.2	0.3	2.1	0.7	1.5	0.2	0.2	0.2	0.3	0.2	2.6	0.7
7-AP avg.	0.1	0.0	0.0	0.0	0.0	0.0	1.5	0.1	0.2	0.0	0.0	0.0	0.0	0.0	1.1	0.1	0.1	0.0	0.0	0.0	0.0	0.0	1.9	0.1
8-AP avg.	0.0	0.0	0.0	0.0	0.0	0.0	0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.9	0.0

Tab. 7: Multichain metaheuristic started from Trivial, Greedy and ROM. 5 seconds given. 1 — 2-opt, 2 — 1DV, 3 — 2DV, 4 — sDV, 5 — 1DV$_2$, 6 — 2DV$_2$, 7 — sDV$_3$, 8 — sDV$_v$.

	Solution error, %																							
	Trivial								Greedy								ROM							
Inst.	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
3gp100	11.8	1.1	1.2	1.1	1.3	1.3	156.9	2.0	5.3	1.2	1.2	1.1	1.4	1.3	5.6	2.2	9.7	1.2	1.2	1.2	1.3	1.3	9.8	2.0
3r150	68.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	41.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	33.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4gp30	3.0	0.9	0.7	0.7	0.8	0.7	0.7	1.1	3.1	0.8	0.7	0.7	1.0	0.7	0.7	1.0	2.1	0.7	0.7	0.7	0.7	0.7	0.8	1.0
4r80	45.3	3.4	1.5	1.4	2.3	1.5	0.4	0.0	38.9	3.3	0.9	0.9	3.1	0.9	0.8	0.0	44.6	2.4	1.0	1.0	2.6	1.1	45.3	0.0
5gp12	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
5r40	26.0	15.3	5.3	5.5	10.8	6.3	516.8	0.0	26.8	14.5	5.3	5.8	10.0	5.0	0.0	0.0	28.3	15.3	6.8	6.8	11.0	7.0	152.5	0.0
6gp8	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4
6r22	8.6	20.0	6.8	4.5	7.7	5.9	0.0	0.0	9.1	20.9	7.7	2.7	7.3	4.5	0.0	0.0	6.4	20.5	8.6	5.5	9.5	5.0	30.0	0.0
7gp5	3.9	3.9	3.5	3.9	3.9	3.9	3.9	3.9	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.9	3.9	3.9	3.9	3.9	3.5	3.9	3.9
7r14	0.0	18.6	7.1	0.7	0.0	2.1	0.0	0.0	0.0	23.6	8.6	1.4	0.7	2.9	0.0	0.0	0.0	21.4	7.1	2.1	0.0	2.1	0.0	0.0
8gp4	2.1	5.2	4.7	4.2	2.1	4.2	3.6	4.2	1.6	3.6	2.6	2.6	2.1	1.6	3.6	3.1	0.5	4.7	5.2	4.2	2.6	4.7	4.7	3.6
8r9	0.0	14.4	2.2	0.0	0.0	0.0	0.0	0.0	0.0	17.8	2.2	0.0	0.0	0.0	0.0	0.0	0.0	14.4	2.2	0.0	0.0	0.0	0.0	0.0
All avg.	14.4	7.2	3.1	2.2	2.7	2.5	57.2	1.3	11.1	7.8	3.0	1.9	2.7	2.0	1.5	1.1	11.1	7.4	3.4	2.4	3.0	2.5	20.9	1.2
GP avg.	4.1	2.5	2.3	2.3	2.0	2.3	28.2	2.5	2.9	2.2	2.0	2.0	2.0	1.8	2.9	2.3	3.4	2.4	2.5	2.3	2.1	2.4	3.8	2.4
Rand. avg.	24.7	11.9	3.8	2.0	3.5	2.6	86.2	0.0	19.4	13.3	4.1	1.8	3.5	2.2	0.1	0.0	18.8	12.3	4.3	2.6	3.9	2.5	38.0	0.0
3-AP avg.	40.0	0.6	0.6	0.6	0.6	0.6	78.5	1.0	23.3	0.6	0.6	0.6	0.7	0.7	2.8	1.1	21.7	0.6	0.6	0.6	0.7	0.7	4.9	1.0
4-AP avg.	24.1	2.1	1.1	1.0	1.5	1.1	0.5	0.6	21.0	2.0	0.8	0.8	2.0	0.8	0.7	0.5	23.3	1.5	0.8	0.8	1.7	0.9	23.0	0.5
5-AP avg.	13.8	8.4	3.4	3.5	6.1	3.9	259.1	0.8	14.1	8.0	3.4	3.6	5.8	3.3	0.8	0.8	14.9	8.4	4.1	4.1	6.3	4.3	77.0	0.8
6-AP avg.	5.5	11.2	4.6	3.5	5.1	4.2	1.2	1.2	5.7	11.7	5.1	2.6	4.8	3.5	1.2	1.2	4.4	11.4	5.5	3.9	6.0	3.7	16.2	1.2
7-AP avg.	2.0	11.2	5.3	2.3	2.0	3.0	2.0	2.0	1.8	13.5	6.0	2.5	2.1	3.2	1.8	1.8	2.0	12.7	5.5	3.0	2.0	2.8	2.0	2.0
8-AP avg.	1.0	9.8	3.5	2.1	1.0	2.1	1.8	2.1	0.8	10.7	2.4	1.3	1.0	0.8	1.8	1.6	0.3	9.6	3.7	2.1	1.3	2.3	2.3	1.8
3cq150	75.2	3.9	3.8	3.7	4.4	4.3	1219.1	491.9	38.2	2.5	2.5	2.5	3.1	3.0	41.1	20.9	36.8	3.9	3.8	3.7	4.8	4.8	36.8	24.2
3g150	0.0	0.0	0.0	0.0	0.0	0.0	865.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	19.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.9	0.0
3p150	0.0	0.0	0.0	0.0	0.0	0.0	76.3	76.3	0.0	0.0	0.0	0.0	0.0	0.0	215.3	215.3	0.0	0.0	0.0	0.0	0.0	0.0	7.2	7.2
3sr150	85.8	4.5	4.3	4.5	5.6	5.5	1249.7	630.5	41.0	3.2	3.2	3.2	3.4	3.3	41.9	7.4	42.8	4.0	3.9	4.0	4.9	4.9	42.8	32.7
4cq50	12.7	2.8	4.4	4.2	3.6	4.8	283.6	9.7	10.6	1.9	2.9	2.9	2.5	3.5	13.9	6.6	13.5	3.8	3.2	3.2	3.8	3.6	28.4	9.6
4g50	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4p50	0.0	0.0	0.0	0.0	0.0	0.0	102.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	484.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	8.3	0.0
4sr50	16.4	3.0	3.4	3.5	3.1	3.9	155.4	10.7	13.7	2.1	3.0	3.1	2.2	3.4	19.5	7.1	15.9	3.8	4.2	4.0	3.9	4.8	29.2	10.5
5cq30	3.4	2.1	1.4	1.4	3.0	1.6	154.5	4.4	3.6	2.0	2.2	2.2	2.3	2.2	20.2	3.2	4.1	2.2	2.2	2.2	2.7	2.3	21.2	4.6
5g30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5p30	0.0	0.0	0.0	0.0	0.0	0.0	137.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1016.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	7.6	0.0
5sr30	6.0	2.4	3.4	3.4	2.7	3.6	195.6	5.3	4.6	2.3	2.3	2.2	2.2	2.4	15.8	4.2	4.7	3.4	3.1	3.2	3.9	3.4	27.6	6.4
6cq18	2.8	2.1	2.0	1.4	1.8	1.8	141.9	3.0	1.9	1.6	1.5	1.5	1.7	1.2	15.4	2.3	2.7	2.2	1.9	1.8	1.4	2.1	18.1	2.3
6g18	0.0	0.0	0.0	0.0	0.0	0.0	260.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	26.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.6	0.0
6p18	0.0	0.0	0.0	0.0	0.0	0.0	162.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2117.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	7.8	0.0
6sr18	3.8	1.9	2.4	2.2	2.3	2.3	120.7	3.5	3.0	2.0	2.1	2.0	2.1	2.1	13.2	2.6	3.9	2.3	1.8	2.1	2.7	1.7	19.1	3.0
7cq12	0.9	1.0	1.0	1.0	0.5	0.9	91.5	1.2	0.7	0.6	0.7	0.6	0.8	0.4	13.8	0.6	1.1	0.8	0.2	0.9	0.8	0.3	14.8	1.1
7g12	0.0	0.0	0.0	0.0	0.0	0.0	156.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	18.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.4	0.0
7p12	0.0	0.0	0.0	0.0	0.0	0.0	346.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3161.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	9.2	0.0
7sr12	1.1	1.4	0.9	1.2	1.0	1.1	77.7	0.9	1.4	1.0	1.1	1.1	0.6	0.7	9.4	1.2	1.8	1.1	0.7	0.8	1.0	1.0	14.9	1.2
8cq8	0.1	0.3	0.2	0.2	0.1	0.4	62.3	0.2	0.2	0.2	0.1	0.2	0.1	0.1	10.4	0.3	0.2	0.3	0.2	0.3	0.2	0.2	9.9	0.3
8g8	0.0	0.0	0.0	0.0	0.0	0.0	104.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	14.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.4	0.0
8p8	0.0	0.0	0.0	0.0	0.0	0.0	176.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3604.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	9.0	0.0
8sr8	0.5	0.2	0.3	0.4	0.1	0.2	51.9	0.3	0.2	0.5	0.4	0.4	0.2	0.3	6.6	0.6	0.3	0.5	0.3	0.6	0.2	0.3	9.8	0.4
All avg.	8.7	1.1	1.1	1.1	1.2	1.3	258.0	51.6	5.0	0.8	0.9	0.9	0.9	0.9	454.3	11.3	5.3	1.2	1.1	1.1	1.3	1.2	13.8	4.3
Clique avg.	15.9	2.0	2.1	2.0	2.2	2.3	325.5	85.1	9.2	1.5	1.7	1.7	1.8	1.7	19.1	5.6	9.7	2.2	1.9	2.0	2.3	2.2	21.5	7.0
Geom. avg.	0.0	0.0	0.0	0.0	0.0	0.0	231.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	13.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.5	0.0
Product avg.	0.0	0.0	0.0	0.0	0.0	0.0	167.0	12.7	0.0	0.0	0.0	0.0	0.0	0.0	1766.7	35.9	0.0	0.0	0.0	0.0	0.0	0.0	8.2	1.2
SR avg.	18.9	2.3	2.5	2.5	2.5	2.8	308.5	108.5	10.7	1.8	2.0	2.0	1.8	2.0	17.7	3.9	11.6	2.5	2.3	2.4	2.8	2.7	23.9	9.0
3-AP avg.	40.2	2.1	2.0	2.1	2.5	2.4	852.6	299.7	19.8	1.4	1.4	1.4	1.6	1.6	79.4	60.9	19.9	2.0	1.9	1.9	2.4	2.4	22.4	16.0
4-AP avg.	7.3	1.5	2.0	1.9	1.7	2.2	135.4	5.1	6.1	1.0	1.5	1.5	1.2	1.7	129.4	3.4	7.4	1.9	1.8	1.8	1.9	2.1	16.5	5.0
5-AP avg.	2.3	1.2	1.2	1.2	1.4	1.3	121.8	2.4	2.1	1.1	1.1	1.1	1.1	1.2	263.8	1.8	2.2	1.4	1.3	1.4	1.6	1.4	14.1	2.8
6-AP avg.	1.6	1.0	1.1	0.9	1.0	1.0	171.4	1.6	1.2	0.9	0.9	0.9	1.0	0.8	543.1	1.2	1.7	1.1	0.9	1.0	1.0	1.0	11.9	1.3
7-AP avg.	0.5	0.6	0.5	0.6	0.4	0.5	167.9	0.5	0.5	0.4	0.4	0.4	0.4	0.3	800.9	0.5	0.7	0.5	0.2	0.4	0.5	0.3	10.3	0.6
8-AP avg.	0.2	0.1	0.1	0.1	0.0	0.2	98.9	0.1	0.1	0.2	0.1	0.2	0.1	0.1	909.1	0.2	0.1	0.2	0.1	0.2	0.1	0.1	7.5	0.2

Tab. 8: Multichain metaheuristic started from Trivial, Greedy and ROM. 10 seconds given. 1 — 2-opt, 2 — 1DV, 3 — 2DV, 4 — sDV, 5 — 1DV$_2$, 6 — 2DV$_2$, 7 — sDV$_3$, 8 — sDV$_v$.

	Solution error, %																							
	Trivial								Greedy								ROM							
Inst.	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
3gp100	11.2	1.0	1.0	1.0	1.1	1.1	2.5	1.7	5.3	1.0	1.0	1.0	1.1	1.1	2.8	1.8	9.7	1.0	0.9	1.0	1.2	1.1	2.3	1.7
3r150	65.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	41.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	33.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4gp30	2.9	0.7	0.7	0.7	0.7	0.7	0.7	1.0	2.6	0.8	0.7	0.7	0.7	0.7	0.7	0.9	2.1	0.7	0.7	0.7	0.7	0.7	0.7	0.8
4r80	43.8	2.5	0.9	0.9	2.1	1.0	0.2	0.0	38.1	2.3	0.6	0.6	2.3	0.6	0.5	0.0	42.4	2.0	0.8	0.8	2.0	0.8	0.5	0.0
5gp12	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
5r40	25.8	12.5	4.0	4.0	9.0	4.0	0.0	0.0	23.5	13.3	4.8	4.5	9.0	4.8	0.0	0.0	25.8	14.3	5.5	5.5	9.8	5.5	0.0	0.0
6gp8	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4
6r22	6.4	18.2	5.9	2.7	6.4	5.0	0.0	0.0	6.4	18.2	6.4	2.3	5.0	2.7	0.0	0.0	5.9	17.3	6.8	3.6	5.5	4.5	0.0	0.0
7gp5	3.9	3.9	3.5	3.9	3.9	3.9	3.1	3.9	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.9	3.9	3.5	3.9	3.9	3.5	3.9	3.9
7r14	0.0	16.4	4.3	0.0	0.0	0.0	0.0	0.0	0.0	19.3	5.0	0.0	0.0	1.4	0.0	0.0	0.0	17.9	5.0	0.0	0.0	0.0	0.0	0.0
8gp4	0.0	5.2	4.7	3.6	1.6	2.6	3.6	4.2	1.0	3.1	2.6	2.1	1.0	1.6	3.6	2.1	0.0	4.7	5.2	3.6	1.6	3.6	4.2	2.6
8r9	0.0	13.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	13.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	10.0	1.1	0.0	0.0	0.0	0.0	0.0
All avg.	13.6	6.5	2.4	1.7	2.4	1.9	1.2	1.2	10.5	6.6	2.4	1.6	2.2	1.7	1.3	1.0	10.6	6.3	2.8	1.9	2.4	2.0	1.3	1.1
GP avg.	3.7	2.4	2.3	2.2	1.9	2.0	2.3	2.4	2.7	2.1	2.0	1.9	1.7	1.8	2.4	2.0	3.3	2.4	2.4	2.2	1.9	2.1	2.5	2.2
Rand. avg.	23.5	10.5	2.5	1.3	2.9	1.7	0.0	0.0	18.2	11.1	2.8	1.2	2.7	1.6	0.1	0.0	17.9	10.2	3.2	1.6	2.9	1.8	0.1	0.0
3-AP avg.	38.2	0.5	0.5	0.5	0.6	0.6	1.3	0.9	23.3	0.5	0.5	0.5	0.6	0.5	1.4	0.9	21.6	0.5	0.5	0.5	0.6	0.5	1.2	0.9
4-AP avg.	23.3	1.6	0.8	0.8	1.4	0.8	0.5	0.5	20.4	1.5	0.7	0.7	1.5	0.7	0.6	0.4	22.2	1.3	0.7	0.7	1.3	0.7	0.6	0.4
5-AP avg.	13.6	7.0	2.8	2.8	5.3	2.8	0.8	0.8	12.5	7.4	3.1	3.0	5.3	3.1	0.8	0.8	13.6	7.9	3.5	3.5	5.6	3.5	0.8	0.8
6-AP avg.	4.4	10.3	4.2	2.6	4.4	3.7	1.2	1.2	4.4	10.3	4.4	2.3	3.7	2.6	1.2	1.2	4.2	9.8	4.6	3.0	3.9	3.5	1.2	1.2
7-AP avg.	2.0	10.2	3.9	2.0	2.0	2.0	1.6	2.0	1.8	11.4	4.3	1.8	1.8	2.5	1.8	1.8	2.0	10.9	4.3	2.0	2.0	1.8	2.0	2.0
8-AP avg.	0.0	9.3	2.3	1.8	0.8	1.3	1.8	2.1	0.5	8.2	1.3	1.0	0.5	0.8	1.8	1.0	0.0	7.3	3.2	1.8	0.8	1.8	2.1	1.3
3cq150	71.4	1.9	1.7	1.8	3.0	2.9	1219.1	8.8	38.2	1.3	1.3	1.3	2.0	2.0	41.1	6.0	36.8	2.4	2.2	2.3	3.0	2.9	36.8	10.1
3g150	0.0	0.0	0.0	0.0	0.0	0.0	865.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	19.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.9	0.0
3p150	0.0	0.0	0.0	0.0	0.0	0.0	76.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	215.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	7.2	0.0
3sr150	82.1	3.0	3.1	2.9	4.0	3.9	1249.7	10.5	41.0	1.9	1.9	1.9	2.8	2.8	41.9	6.2	42.8	2.9	2.8	2.8	3.7	3.7	42.8	10.3
4cq50	11.1	2.6	3.4	3.3	3.2	3.8	11.3	7.8	10.3	1.5	2.7	2.7	2.4	2.8	8.6	4.6	11.1	2.8	2.8	2.8	3.5	3.1	12.3	7.3
4g50	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4p50	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4sr50	14.6	2.0	2.7	2.6	2.6	3.2	12.9	8.1	12.5	2.1	2.3	2.3	2.0	2.8	9.1	5.0	14.7	3.6	3.4	3.4	3.3	3.5	12.6	7.8
5cq30	2.2	2.1	1.3	1.3	2.4	1.3	8.0	3.5	2.8	1.5	2.2	2.2	1.8	2.2	5.0	2.6	3.0	2.2	2.0	2.2	2.5	2.2	8.0	4.2
5g30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5p30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	809.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5sr30	4.6	2.2	3.3	3.3	2.6	3.4	9.6	4.3	3.6	1.9	1.9	2.0	2.0	2.0	6.6	3.2	3.5	3.1	2.9	2.9	3.6	2.9	13.7	4.6
6cq18	2.8	2.1	2.0	1.4	1.8	1.7	5.6	2.4	1.9	1.6	1.5	1.5	1.5	1.2	4.2	1.9	2.0	2.2	1.9	1.8	1.4	1.9	5.8	2.3
6g18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6p18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1038.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6sr18	3.5	1.8	2.4	2.0	2.3	2.3	6.5	3.2	3.0	2.0	2.1	1.9	2.1	2.1	5.3	2.3	3.9	2.3	1.8	2.1	2.6	1.7	6.7	2.6
7cq12	0.9	1.0	1.0	1.0	0.5	0.9	38.6	1.2	0.7	0.6	0.7	0.6	0.8	0.4	7.7	0.4	1.0	0.8	0.2	0.8	0.8	0.3	9.2	0.9
7g12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7p12	0.0	0.0	0.0	0.0	0.0	0.0	346.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3161.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	9.2	0.0
7sr12	1.0	1.4	0.9	1.1	1.0	1.1	62.4	0.9	1.0	1.0	1.1	1.1	0.6	0.7	9.4	1.2	1.6	1.1	0.7	0.8	1.0	0.9	13.0	1.1
8cq8	0.1	0.2	0.2	0.2	0.0	0.4	62.3	0.2	0.2	0.2	0.1	0.2	0.1	0.1	10.4	0.3	0.2	0.3	0.2	0.3	0.2	0.2	9.9	0.3
8g8	0.0	0.0	0.0	0.0	0.0	0.0	104.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	14.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.4	0.0
8p8	0.0	0.0	0.0	0.0	0.0	0.0	176.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3604.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	9.0	0.0
8sr8	0.5	0.2	0.3	0.4	0.1	0.2	51.9	0.2	0.2	0.5	0.4	0.4	0.2	0.3	6.6	0.6	0.3	0.5	0.3	0.6	0.1	0.3	9.8	0.4
All avg.	8.1	0.9	0.9	0.9	1.0	1.0	179.4	2.1	4.8	0.7	0.8	0.8	0.8	0.8	375.8	1.4	5.0	1.0	0.9	0.9	1.1	1.0	8.8	2.2
Clique avg.	14.8	1.7	1.6	1.5	1.8	1.8	224.1	4.0	9.0	1.1	1.4	1.4	1.4	1.5	12.8	2.6	9.0	1.8	1.6	1.7	1.9	1.8	13.7	4.2
Geom. avg.	0.0	0.0	0.0	0.0	0.0	0.0	161.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.7	0.0
Product avg.	0.0	0.0	0.0	0.0	0.0	0.0	99.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1471.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.2	0.0
SR avg.	17.7	1.8	2.1	2.0	2.1	2.4	232.2	4.5	10.2	1.6	1.6	1.6	1.6	1.8	13.2	3.1	11.1	2.2	2.0	2.1	2.4	2.2	16.4	4.5
3-AP avg.	38.4	1.2	1.2	1.2	1.7	1.7	852.6	4.8	19.8	0.8	0.8	0.8	1.2	1.2	79.4	3.1	19.9	1.3	1.3	1.3	1.7	1.6	22.4	5.1
4-AP avg.	6.4	1.2	1.5	1.5	1.4	1.7	6.1	4.0	5.7	0.9	1.2	1.2	1.1	1.4	4.4	2.4	6.4	1.6	1.6	1.6	1.7	1.7	6.2	3.8
5-AP avg.	1.7	1.1	1.1	1.1	1.3	1.2	4.4	1.9	1.6	0.8	1.0	1.1	1.0	1.1	205.3	1.5	1.6	1.3	1.2	1.3	1.5	1.3	5.4	2.2
6-AP avg.	1.6	1.0	1.1	0.9	1.0	1.0	3.0	1.4	1.2	0.9	0.9	0.9	0.9	0.8	262.0	1.0	1.5	1.1	0.9	1.0	1.0	0.9	3.1	1.2
7-AP avg.	0.5	0.6	0.5	0.5	0.4	0.5	111.8	0.5	0.4	0.4	0.4	0.4	0.4	0.3	794.7	0.4	0.6	0.5	0.2	0.4	0.5	0.3	7.8	0.5
8-AP avg.	0.2	0.1	0.1	0.1	0.0	0.2	98.9	0.1	0.1	0.2	0.1	0.2	0.1	0.1	909.1	0.2	0.1	0.2	0.1	0.2	0.1	0.1	7.5	0.2

Tab. 9: Heuristics comparison for the instances with independent weights.

Inst.	< 10 ms	< 30 ms	< 100 ms	< 300 ms	< 1000 ms
3r150	—	**C** sDV Gr 1.4 **C** sDV 1.5 **C** 2DV_2 1.5	**C** $s\text{DV}_\text{v}$ 0.3	**C** sDV 0.0 **C** 2DV_2 0.0 **C** $s\text{DV}_\text{v}$ 0.0 **C** $s\text{DV}_\text{v}$ Gr 0.0 **C** sDV R 0.0 **C** 2DV_2 R 0.0 **C** $s\text{DV}_\text{v}$ R 0.0	(no better solutions)
4r80	**C** 1DV 25.8	sDV Gr 6.1 2DV_2 Gr 6.1	$s\text{DV}_\text{v}$ Gr 1.5	**C** $s\text{DV}_\text{v}$ Gr 0.3	**C** $s\text{DV}_\text{v}$ 0.0 **C** $s\text{DV}_\text{v}$ Gr 0.0 **C** $s\text{DV}_\text{v}$ R 0.0
5r40	1DV_2 Gr 15.0	2DV Gr 13.5 sDV Gr 13.5 2DV_2 Gr 13.5	**C** $s\text{DV}_\text{v}$ 1.2	**C** $s\text{DV}_\text{v}$ 0.0	(no better solutions)
6r22	**C** 2DV 46.4 **C** sDV 47.3	2-opt Gr 25.9	**C** $s\text{DV}_\text{v}$ Gr 1.4	**C** $s\text{DV}_\text{v}$ Gr 0.0	(no better solutions)
7r14	**C** 2-opt Gr 28.6	**C** $s\text{DV}_\text{v}$ Gr 13.6	**C** $s\text{DV}_\text{v}$ 1.4	**C** $s\text{DV}_\text{v}$ 0.0 **MC** $s\text{DV}_\text{v}$ 0.0 **C** $s\text{DV}_\text{v}$ Gr 0.0	(no better solutions)
8r9	**C** 2-opt Gr 22.2 **C** 2-opt 24.4	**C** $s\text{DV}_\text{v}$ 12.2	**C** $s\text{DV}_\text{v}$ 0.0	(no better solutions)	(no better solutions)
Total	—	**C** sDV Gr 18.6 **C** 2DV_2 Gr 19.3 **C** sDV 20.2	**C** $s\text{DV}_\text{v}$ Gr 4.8	**C** $s\text{DV}_\text{v}$ Gr 0.1	**C** $s\text{DV}_\text{v}$ 0.0 **C** $s\text{DV}_\text{v}$ Gr 0.0

Tab. 10: Heuristics comparison for the instances with decomposable weights.

Inst.	< 100 ms	< 300 ms	< 1000 ms	< 3000 ms	< 10000 ms
3cq150	sDV Gr 8.1	**C** sDV Gr 7.8 **C** 2DV$_2$ Gr 7.8	**MC** sDV Gr 6.6 **MC** 2DV$_2$ Gr 7.1	**MC** sDV Gr 3.1 **MC** 2DV$_2$ Gr 3.4	**MC** sDV Gr 1.3
3sr150	**C** sDV Gr 9.6 sDV Gr 9.8 1DV$_2$ Gr 9.8 **C** 2DV$_2$ Gr 10.2	**C** sDV Gr 8.4 **C** 2DV$_2$ Gr 8.4	**MC** sDV Gr 6.6	**MC** sDV Gr 3.5	**MC** sDV Gr 2.0
4cq50	**C** 1DV 9.7 **MC** 1DV 10.0 **C** 1DV$_2$ 10.3	**MC** 1DV Gr 6.4 **MC** 1DV 6.9	**MC** 1DV Gr 4.7 **MC** 1DV$_2$ Gr 4.9 **MC** 1DV 5.0 **MC** sDV R 5.1 **MC** 1DV$_2$ R 5.1	**MC** 1DV Gr 2.7	**MC** 1DV Gr 1.5
4sr50	**C** 1DV 11.7 **MC** 1DV 12.2	**MC** 1DV Gr 7.0 **MC** 1DV 7.7	**MC** 1DV Gr 4.7 **MC** 1DV$_2$ Gr 5.0	**MC** 1DV Gr 2.6 **MC** 1DV$_2$ Gr 2.7	**MC** 1DV$_2$ Gr 2.0 **MC** 1DV 2.0 **MC** 1DV Gr 2.1 **MC** 1DV M-R 2.1
5cq30	**C** 1DV 6.3 **MC** 1DV 6.4	**MC** 1DV 3.2	**MC** 2DV 2.6 **MC** 1DV 2.6 **MC** sDV 2.7	**MC** 2DV 1.7 **MC** sDV 1.7	**MC** sDV 1.3 **MC** 2DV 1.3 **MC** 2DV$_2$ 1.3
5sr30	**MC** 1DV 7.9 **C** 1DV 8.3	**MC** 1DV 3.9	**MC** 1DV 3.2	**MC** 1DV$_2$ Gr 2.4 **MC** 2DV Gr 2.5 **MC** sDV Gr 2.5 **MC** 1DV 2.5 **MC** 2DV$_2$ Gr 2.6	**MC** 2DV Gr 1.9 **MC** 1DV Gr 1.9 **MC** sDV Gr 2.0 **MC** 2DV$_2$ Gr 2.0 **MC** 1DV$_2$ Gr 2.0
6cq18	**C** 1DV 2.1	**C** 1DV 1.0	**C** 1DV 0.7	**C** 2DV Gr 0.3	**C** sDV Gr 0.0
6sr18	**MC** 1DV 3.8 **C** 1DV 3.8	**MC** 1DV 2.1 **C** 1DV 2.1	**C** 2DV 1.4 **C** 2DV$_2$ R 1.5	**C** 1DV 0.8	**C** sDV Gr 0.3
7cq12	**C** 1DV 0.7	**C** 1DV 0.2	**C** 1DV$_2$ 0.1	**C** 1DV 0.0	**C** 1DV 0.0 **C** 2DV$_2$ 0.0 **C** 1DV Gr 0.0 **C** 1DV$_2$ Gr 0.0 **C** 1DV R 0.0 **C** 2DV R 0.0 **C** sDV R 0.0 **C** 2DV$_2$ R 0.0
7sr12	**C** 1DV 1.2	**C** 1DV 0.5 **C** 1DV$_2$ 0.5	**C** 1DV R 0.1	**C** 2DV 0.0	(no better solutions)
8cq8	**C** 1DV 0.0	**C** 1DV 0.0	(no better solutions)	(no better solutions)	(no better solutions)
8sr8	**C** 1DV 0.3	**C** 1DV 0.0 **C** 2DV 0.0	**C** 1DV 0.0 **C** 2DV 0.0 **C** 1DV$_2$ 0.0 **C** 2DV$_2$ 0.0 **C** 1DV Gr 0.0 **C** 1DV$_2$ Gr 0.0 **C** 2DV$_2$ Gr 0.0 **C** 2-opt R 0.0 **C** 1DV R 0.0 **C** 2DV R 0.0 **C** 1DV$_2$ R 0.0 **C** 2DV$_2$ R 0.0	(no better solutions)	(no better solutions)
Total	**C** 1DV 6.4	**C** 1DV 4.5 **C** 2DV 5.0	**MC** 1DV 3.5 **MC** 2DV 3.7 **C** 2DV 3.7 **MC** 1DV R 3.8	**MC** 1DV Gr 1.9 **MC** 2DV Gr 2.1 **MC** sDV Gr 2.1 **MC** 1DV$_2$ Gr 2.1	**MC** 1DV Gr 1.3