Adaptive Multi-operator MetaHeuristics for quadratic assignment problems

Madalina M. Drugan[1] and El-Ghazali Talbi[2]

[1] Artificial Intelligence lab, Vrije Universiteit Brussels, Pleinlaan 2, 1050-Brussels, Belgium
[2] Univ. Lille, CNRS and INRIA Lille Nord Europe Parc Scientifique de la Haute Borne 40, avenue Halley Bt.A, Park Plaza 59650 Villeneuve d'Ascq, France

Abstract. Local search based algorithms are a general and computational efficient metaheuristic. Restarting strategies are used in order to not be stuck in a local optimum. Iterated local search restarts the local search using perturbator operators, and the variable neighbourhood search alternates local search with various neighbourhoods. These two popular restarting techniques, or operators, evolve independently and are disconnected. We propose a metaheuristic framework, we call it *multi-operator metaheuristics*, which allows the alternative or simultaneously usage of the two restarting methods.
Tuning the parameters, i.e. the neighbourhood size and the perturbation rate, is essential for the performance of metaheuristics. We automatically adapt the parameters for the two restarting operators using variants of *adaptive pursuit* for the multi-operators metaheuristic algorithms.
We experimentally study the performance of several instances of the new class of metaheuristics on the quadratic assignment problem (QAP) instances, a well-known and difficult combinatorial optimization problem.

1 Introduction

Metaheuristics (Talbi, 2009; Hoos and Stutzle, 2005) is a general, successful and powerful search method for difficult optimization problems. Local search (LS) based metaheuristics starts from an initial solution and iteratively generates new solutions using a neighbourhood strategy. Each step, a solution that improves over the existing best-so-far solution is chosen. The local search stops when there is no possible improvement, i.e. in a local optimum. Because LS can be stuck in local optima, some advanced local search algorithms consist in restarting the LS.

We consider two basic techniques to restart the local search. *Variable neighbourhood search* (VNS) (Mladenovic and Hansen, 1997; Hansen et al., 2008) is a variant of local search that changes the neighbourhood function to escape local optimum. A set of neighbourhood functions are alternated either in a predefined or uniform randomly order.

Multi-restart local search (MLS) restarts local search multiple times from uniform randomly chosen initial solutions in order to find different basin of attractions in different part of the search space. There are certain limitations in the design of multi-restart LS because it is basically random sampling in the space of local optima, it does not scale up for large number of local optima.

To improve upon multi-restart LS's performance, stochastic local search aims to escape from local optimal sets by stochastic perturbation operators that preserve partial information of the perturbed solutions. In case of *iterated LS* (ILS) (Hoos and Stutzle, 2005), the perturbation operator is mutation. It has been pointed out (Drugan and Thierens, 2010; Drugan and Thierens, 2012) that the experimental performance is sensitive to the choice of the mutation' parameters.

The two different techniques restart the local search using two different parameters: i) the neighbourhood functions for VNS, and ii) the mutation rate for the iterated LS. Section 2 gives some background on metaheuristic algorithms and we show a detailed example of metaheuristics for the quadratic assignment problem (QAP).

Quadratic assignment problem (QAP). Intuitively, QAPs can be described as the optimal assignment of a number of facilities to a number of locations. QAPs are NP-hard combinatorial optimization problems that model many real-world problems (i.e., scheduling, vehicle routing). Let us consider N facilities, a set $\Pi(N)$ of all permutations of $\{1, \ldots, N\}$ and the $N \times N$ distance matrix $A = (a_{ij})$, where a_{ij} is the distance between location i and location j. We assume a flow matrix $B = (b_{ij})$ where b_{ij} represents the flow from facility i to facility j. The goal is to minimize the *cost function*

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

where $\pi(\cdot)$ is a permutation. It takes quadratic time to evaluate this function.

The main contributions. In general, certain metaheuristic algorithm instances are preferred to solve specific problem instances. For example, iterated local search is commonly used to optimize QAP instances, but seldom used with variable size neighbourhood. There is no study to indicate when and for which instances one of the two methods is superior and, even more, if the alternative or simultaneously use of the two restarting methods is beneficial or not.

We propose a class of metaheuristic that allows the use of both operators to restart the search: i) variable neighbourhood search, and ii) stochastic perturbator operators. We call this the *Multi-operator MetaHeuristics* (MMH) problem and we introduce it in Section 3.

We assume that these two restarting techniques are complementary but with the same goal of finding better local optimum. The stochastic perturbator operators exploit the structure of the search space, and thus they can be considered the *diversification* technique of LS. The variable neighbourhood search is the *intensification* technique for the MMH problem because it explores the search space by changing the size of the neighbourhood.

The multi-operator metaheuristic problem has several parameters to tune. The parameters of the variable neighbourhood search are the size of the neighbourhoods, and we need a strategy to decide when and how to alternate the neighbours. The iterated local search has similar parameters that are the size of the stochastic perturbator operator(s). Again, we need a strategy to decide which mutation rate to use. In addition, an adaptive multi-operator metaheuristic algorithm needs a strategy to decide when and how to use each restarting method.

Tuning the parameters, and in particular tuning parameters of the restarting strategies for metaheuristics, is a complex process that is preferable to carry out automatically. *Pursuit allocation strategies* (AP) (Thierens, 2005) are on-line operator selection algorithms that adapt a selection probability distribution such that the operator with the maximal estimated reward is often pursued. The second contribution of this paper is the generalization of the adaptive pursuit algorithm to adapt more than one parameter, e.g. the size of the neighbourhood and the mutation step. We call this *multi-operator adaptive pursuit* and introduce it in Section 4. We assume that the two parameters are correlated and their alternatively or simultaneously usage helps the exploration of the search space. Although we apply this type of algorithms on multi-restart adaptive metaheuristics, the scope of the multi-operator adaptive pursuit is broader. For example, it could be used to simultaneously adapt the mutation and recombination operators of evolutionary algorithms, or any other set of operators of an evolutionary algorithm.

In Section 5, we show preliminary experimental results of the proposed algorithms on several instances of the *quadratic assignment problem* (QAP) (Loiola et al., 2007; Drugan, 2014). Section 6 concludes the paper.

2 Metaheuristics: background

In this section, we present background knowledge on two, up to date, independent metaheuristic algorithms. Iterated Local search (ILS) (Stützle, 2006) is a very popular local search based metaheuristic because of its simplicity and the ease in usage. Variable Neighbourhood search (VNS) (Mladenovic and Hansen, 1997) is one of the first metaheuristic popular in solving specific type of problems like the min-max combinatorial optimization problems. We show an effective implementation of local search for QAP instances, which is a permutation problem.

The common part of the two metaheuristics is the local search function. Consider that local search as a combination of: i) a neighbourhood function, and ii) a neighbourhood exploration strategy, or an improvement strategy.

The neighbourhood function, $\mathcal{N}$, generates solutions in the neighbourhood of a given solution s. $\mathcal{N}(s)$ has as input a solution s and returns the set of neighbours for that solution. Thus, $\mathcal{N} : \mathcal{S} \leftarrow \mathcal{P}(\mathcal{S})$ is a function that maps the solution space $\mathcal{S}$ to sets of solutions $\mathcal{P}(\mathcal{S})$. The neighbourhood function depends on the problem (e.g., quadratic assignment problem) is applied on.

A suitable neighbourhood function for QAPs is the 2-exchange swapping operator that swaps two different facilities. This operator is attractive because of its linear time to compute the difference in the cost function with the condition that the flow and distance matrices are symmetrical.

The size of a neighbourhood increases quadratically with the number of facilities. This means that for a 2-exchange swapping operator, there are $\binom{N}{2}$ neighbours for a solution, and for a m-exchange swapping operator, there are $\binom{N}{m}$ for a single solution. We call this the m-th *neighbourhood exchange rate*.

The neighbourhood exploration strategy, $\mathcal{I}$, decides when to stop the expansion of a neighbourhood $\mathcal{N}(s)$. We consider $\mathcal{I}(s, \mathcal{N})$ a tuple of solutions and neighbourhood

$t \leftarrow 0$;
$s' \leftarrow$ GenerateInitialSolution ;
$s^{(0)} \leftarrow$ LocalSearch(s') ;
while *the stopping criteria is NOT met* **do**
  $s' \leftarrow$ Perturbation$(s^{(t)})$;
  $s'' \leftarrow$ LocalSearch(s') ;
  if s'' *is an improvement over* $s^{(t)}$ **then** $s^{(t+1)} \leftarrow s''$;
  $t \leftarrow t + 1$
end
return *The local optimum solution* $s^{(t)}$

Algorithm 1: Iterated local search (ILS)

functions. There are mainly two neighbourhood exploration techniques (Hansen and Mladenovic, 2006): i) the first improvement, and ii) the best improvement. The *best improvement* explores *all* the individuals in the neighbourhood of a solution and selects the best solution that improves over the initial solution of a neighbourhood. The *first improvement* stops when the first improvement to the initial solution is found.

Note that the best improvement strategy does not depend on the problem because the entire neighbourhood is expanded. The first improvement strategy depends on the definition of the improvement (Drugan and Thierens, 2012).

For the QAP problems, which is a minimization problem, a solution s is considered better than another solution s' iff $c(s) < c(s')$. The cost function to optimize is $c(\cdot)$, where $c : \mathcal{S} \leftarrow \mathbb{R}$, and $\mathcal{S}$ is the solution space and $\mathbb{R}$ is the real valued space.

Furthermore, for large neighbourhoods, the first improvement strategies are more efficient than the best improvement strategies which will spend considerable amount of times in the neighbourhood of the first, suboptimal solutions. Also experimentally, the first improvement based metaheuristics are acknowledged to outperform the best improvement metaheuristics for some combinatorial optimization problems (Hansen and Mladenovic, 2006; Drugan and Thierens, 2012).

2.1 Iterated local search (ILS)

Iterator local search (Stützle, 2006) restarts local search from solutions generated with perturbator operators, like the mutation operator, in order to escape the local optimum. The neighbourhood function, $\mathcal{N}$, is assumed to be fixed, e.g. for the QAP instances is a 2-exchange operator. The pseudo-code for iterated local search is presented in Algorithm 1.

The ILS algorithm starts with a uniform randomly generated solution $s^{(0)}$. A local search function is called on this randomly generated solution, LocalSearch$(s^{(0)})$. With a standard ILS algorithm, this is the only local search call started from a uniform randomly generated solution. Until a stopping criteria is met, we perform the following loop which has a counter t. A new solution s' is generated from the current local optimum solution $s^{(t)}$ using a perturbator operator. We assume that the generated solution is a sub-optimal solution. The local search is restarted from the new solution and the

returned solution s'' replace the current local optimum $s^{(t)}$ iff it is better than $s^{(t)}$. The algorithm returns the best found so far local optimum.

The perturbator operator. Like the neighbourhood function, the perturbator operator depends on the problem instance. In permutation problems like QAPs, the *mutation operator* exchanges facilities between different positions. The m-exchange mutation uniform randomly selects m distinct pairs of positions which are sequentially exchanged.

We assume a mutation exchange rate that is larger than the neighbourhood exchange rate. When LS uses the 2-exchange operator to generate a neighbourhood, the mutation operator should exchange *at least* 3 facilities to escape from the region of attraction of a local optimum.

In practice, this mutation operator is often tuned. A small m could mean that the local search cannot escape the basin of attraction. A large m means basically random sampling in the search space and then the structure of the search space cannot be exploited. The optimal value for m depends on the landscape of the search space (Drugan and Thierens, 2010), i.e. the size of the basin of attractions.

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

The acceptance criteria. The solution returned by local search s'' is accepted, iff it improves the current local optimum solution, $s^{(t)}$. Thus, if a solution is accepted, then $s^{(t)} \leftarrow s''$. Recall that for the QAP instances, a solution s'' is better than $s^{(t)}$ iff $c(s'') < c(s^{(t)})$.

2.2 Variable neighbourhood search (VNS)

VNS (Hansen et al., 2008) is a metaheuristic that systematically change its neighbourhood function in order to escape from the current local optimal solution. A VNS algorithm considers that two neighbourhood functions started from the same solution can have different local optimal solutions. The global optimal solution is considered the best local optimal solution of all neighbourhood functions.

Local search. The definition of local search for VNS generalizes the previous definition of local search from Section 2.1 because it uses a set of neighbourhood functions that are alternated during a local search run. The pseudo-code for the local search algorithm is given in Algorithm 2.

The input for the local search requires an initial solution s and a set of P neighbourhood functions $\mathcal{N} = (\mathcal{N}_1, \ldots, \mathcal{N}_P)$. For the local search function from Section 2.1, we have that the set of neighbourhood functions contains only one neighbourhood functions. We use a counter t for the number of neighbourhood exploration functions. Until an improvement is still possible, we select a neighbourhood function, $\mathcal{N}^{(t)} \in (\mathcal{N}_1, \ldots, \mathcal{N}_P)$, from the set of neighbourhoods in order to explore the corresponding

$t \leftarrow 0$;
while *An improvement is possible* **do**
  $\mathcal{N}^{(t)} \leftarrow \mathsf{SelectNeighbourhood}(\mathcal{N}_1, \ldots, \mathcal{N}_P)$;
  $s' \leftarrow \mathcal{I}(s^{(t)}, \mathcal{N}^{(t)})$;
  if s' *is an improvement over* $s^{(t)}$ **then** $s^{(t)} \leftarrow s'$;
  $t \leftarrow t + 1$
end
return *The local optimum solution* $s^{(t)}$

Algorithm 2: Local search $\mathsf{LS}(s, (\mathcal{N}_1, \ldots, \mathcal{N}_P))$

neighbourhood of the current solution $s^{(t)}$. If the resulting solution s' is an improvement over $s^{(t)}$, the current solution is replaced by s' and the counter t is updated.

SelectNeighbourhood. There are various techniques to alternate the neighbourhood functions (Hansen et al., 2008). The most popular VNS variant is the *variable neighbourhood descend* that deterministically changes the size of the neighbourhood and performs a local search with all neighbourhood functions. The resulting local optimum is the optimum for all the neighbourhood functions. The *reduced VNS* changes uniform randomly the neighbourhood functions each time the exploration neighbourhood function is called. The *skewed VNS* variant resemble, in some extend, the multi-restart LS because it explores regions that are far from the initial solution by generating random restarting solutions.

3 Multi - operator metaheuristics

In this section, we propose a class of metaheuristic algorithms that use two (possible more) schemes that restart the local search. This algorithm is a combination of two metaheuristics, i.e. the iterated local search and the variable neighbourhood search. ILS and VNS have in common the local search function call, but have different restarting strategies. The two restarting strategies can be used simultaneously or alternatively. If the two strategies are used alternatively, another strategy to decide the restarting technique at a time step is needed.

This hybrid types of metaheuristics are motivated by the QAP problems. For the QAP using a neighborhood based on the exchange operator, the cost of exploring local search is quadratic with the size of the neighbourhood. Thus, it is computational inefficient to use neighbourhood functions of large size like required by VNS algorithms. The computational cost of the mutation operator, however, does not change much with the exchange rate. This unbalance between the computational cost of the two techniques for the QAP problem makes the study of the alternation between the two techniques relevant for this class of problems.

3.1 A baseline algorithm

In this setting both restarting strategies can be used simultaneously. The pseudo-code for this algorithm is given in Algorithm 3.

At the initialization, a solution s' is generated uniform randomly with GenerateInitialSolution. The local search function LocalSearch is initialized with the initial solution s' and with the set of neighbourhood strategies $(\mathcal{N}_1, \ldots, \mathcal{N}_P)$. The solution returned by local search is an initial local optimum $s^{(0)}$.

Each iteration, until a stopping criteria is met, a new restarting solution s' is proposed with a perturbator operator over the current local optimum solution $s^{(t)}$. The local search is restarted from this newly generated solution s' and using the set of neighbourhood functions $\mathcal{N}$. The new local optimum $s^{(t+1)}$ improves or it is equal with the current local optimum $s^{(t)}$. The counter of iterations is updated $t \leftarrow t + 1$.

Remark 3.1. We assume here a small number of neighbourhoods in the neighbourhood set $\mathcal{N}$. Then, the local optimum resulted from the local search function is probable not the global optimum and the search needs to be restarted from another region of the search space.

Remark 3.2. The m-exchange rate of the mutation operator needs to be larger than the largest neighbourhood from the neighbourhood set in order to escape the basin of attraction of the current local optimum. A smaller m-exchange rate still make sense with the first improvement strategy where the search is stopped at the first improvement.

Remark 3.3. If the perturbator operator is a uniform random generator, then we have a skewed VNS because the local search is uniform randomly restarted in the search space. If there is only one neighbourhood function in the neighbourhood set, then we have a standard ILS search like in Algorithm 1.

3.2 Selecting one restarting strategy

The baseline algorithm can be quite computational demanding especially for the QAP problem where the size of the neighbourhoods increase quadratically. We propose a variant of the multi-restart strategies metaheuristics where the restarting techniques are alternated rather then simultaneously used.

At the initialization, an uniform random solution is generated s' and a neighbourhood strategy is uniform randomly selected $\mathcal{N}^{(0)}$. The local search LocalSearch$(s', \mathcal{N}^{(0)})$ returns the first local optimum.

Each iteration, until a stopping criteria is met, a restarting strategy is chosen using a function SelectRestartStategy. This strategy can be simply a uniform generator, and then the two restart strategies are uniform randomly selected.

$t \leftarrow 0$;
$s' \leftarrow$ GenerateInitialSolution ;
$s^{(0)} \leftarrow$ LocalSearch$(s', (\mathcal{N}_1, \ldots, \mathcal{N}_P))$;
while *The stopping criteria is NOT met* **do**
  $s' \leftarrow$ Perturbation$(s^{(t)})$;
  $s^{(t+1)} \leftarrow$ LocalSearch$(s', (\mathcal{N}_1, \ldots, \mathcal{N}_P))$;
  $t \leftarrow t + 1$
end
return *The local optimum solution* $s^{(t)}$

Algorithm 3: Multi - operator MetaHeuristics (MMH)

$t \leftarrow 0$;
$s' \leftarrow$ GenerateInitialSolution ;
$\mathcal{N}^{(0)} \leftarrow$ SelectInitialNeighbourhood$(\mathcal{N}_1, \ldots, \mathcal{N}_P)$;
$s^{(0)} \leftarrow$ LocalSearch$(s', \mathcal{N}^{(0)})$;
while *the stopping criteria is NOT met* **do**
  if *SelectRestartStategy is mutation* **then**
    $s' \leftarrow$ Perturbation$(s^{(t)})$;
    $s^{(t+1)} \leftarrow$ LocalSearch$(s', \mathcal{N}_1)$
  else
    $\mathcal{N}^{(t)} \leftarrow$ SelectNeighbourhood$(\mathcal{N}_1, \ldots, \mathcal{N}_P)$;
    $s^{(t+1)} \leftarrow$ LocalSearch$(s^{(t)}, \mathcal{N}^{(t)})$
  end
  $t \leftarrow t + 1$
end
return *The local optimum solution* $s^{(t)}$

Algorithm 4: Alternating operators MetaHeuristics (AMH)

If the selected strategy is the mutation operator, then a (probable suboptimal) solution is generated with the perturbator function in order to restart local search. Like in ILS, the local search will now use a small neighbourhood, where we assume that $\mathcal{N}_1$ is the smallest neighbourhood from the set $\mathcal{N}$. If the variable neighbourhood search is selected then only the neighbourhood is selected with SelectNeighbourhood and the local search is restarted from the current local optimal solution $s^{(t)}$.

Remark 3.4. If only one of the restarting strategies is used, this algorithm is a standard ILS or a standard VNS, after case.

3.3 Setting-up the parameters

The two restart strategies have two operators whose parameters need to be set: i) the mutation operator, and ii) the set of neighbourhood functions. For each operator, we consider a fixed set of parameters.

The set of mutation parameters. For the mutation operator, we consider K parameters, where $\mathcal{M} = (\mathcal{M}_1, \ldots, \mathcal{M}_K)$. A mutation operator that uniform randomly select each parameter from the set $\mathcal{M}$ is shown to outperform the ILS algorithms that use a mutation operator with a single parameter from $\mathcal{M}$.

For the QAP instances, we consider each mutation exchange rate to be a mutation parameter.

Adaptive operator selection. We correlate the usage of a specific mutation operator with the improvement resulted by applying this mutation operator on the current solution. The adaptive pursuit (AP) strategy (Thierens, 2005) is often used for adaptive operator selection. The mutation operator for which the restarted local search improves the most the current solution, will be pursuit often.

The set of neighbourhood function. We assume a fixed set of P neighbourhood functions, $\mathcal{N} = (\mathcal{N}_1, \ldots, \mathcal{N}_P)$. The problem of selecting one neighbourhood function from a set of neighbourhood functions is similar to selecting a perturbator operator from

a set of perturbator operators. Thus, we could uniform randomly select these operators or we could adaptively select the neighbourhoods that improve the most the current solution.

Remark 3.5. In general, in Evolutionary Computation, the parameters of different operators are selected independently. For our problem, the parameters from the mutation set are independently selected from the parameters of the neighbourhood set.

In the next section, we propose a new adaptive operator selection algorithm that consider that the performance of the two operators is correlated and thus that their selection mechanisms should be correlated.

4 Adaptive Multi-operator MetaHeuristics

In this section, we consider an adaptive version of the multi-operator metaheuristics that includes an extension of the adaptive pursuit strategy (Thierens, 2005; Drugan and Thierens, 2011) for multi-operator selection. The two operators can be adapted separately, independently or simultaneously.

The standard *adaptive pursuit* (AP) algorithm adapts a probability vector $\mathcal{P}^{(t)}$ such that the operator that has the highest estimated reward is chosen with very high probability. The target distribution does not change in time and the target distribution is associated with a step-like distribution where one operator has a very large selection probability whereas the probability of selecting the rest of the operators is much smaller.

The biggest difference between the single and the multi-operator adaptive pursuit is that now the rewards are vectors, $\mathbf{R}^{(t)}$, instead of values. Consequently, the temporal probabilities vectors, i.e. $\mathbf{P}^{(t)}$ and $\mathbf{Q}^{(t)}$, are also matrices. Consider the mutation and the neighbourhood operators, each operator with K and P parameters as before. The probability distribution for the mutation operator and for the neighbourhood operator is

$$\mathbf{P}_{\mathcal{N}}^{(t)} = \{\mathbf{P}_{11}^{(t)}, \ldots, \mathbf{P}_{1K}^{(t)}\}, \quad \mathbf{P}_{\mathcal{M}}^{(t)} = \{\mathbf{P}_{21}^{(t)}, \ldots, \mathbf{P}_{2P}^{(t)}\}$$

where $\forall t$, and $\forall j$, and $0 \leq \mathcal{P}_{1j}^{(t)} \leq 1$, $\sum_{j=1}^{K} \mathcal{P}_{1j}^{(t)} = 1$, and $\forall t$, and $\forall j$, and $0 \leq \mathcal{P}_{2j}^{(t)} \leq 1$, $\sum_{j=1}^{P} \mathcal{P}_{1j}^{(t)} = 1$. The quality distribution (or the estimated reward) for the mutation operator and for the neighbourhood operator is

$$\mathbf{Q}_{\mathcal{N}}^{(t)} = \{\mathbf{Q}_{11}^{(t)}, \ldots, \mathbf{Q}_{1K}^{(t)}\}, \quad \mathbf{Q}_{\mathcal{M}}^{(t)} = \{\mathbf{Q}_{21}^{(t)}, \ldots, \mathbf{Q}_{2P}^{(t)}\}$$

4.1 A baseline algorithm

We consider the multi-operator adaptive pursuit algorithm where the probability of selecting a parameter is independently adapted for each operator. Algorithm 5 represents the pseudo-code of this algorithm.

InitializeAdaptationVectors. In the initialization step, $t \leftarrow 0$ and the vectors of each parameter are initialized separately. For the mutation operator, we have $\mathcal{P}_{1j}^{(t)} \leftarrow 1/K$ such that $\sum_{i=1}^{K} \mathcal{P}_{1j}^{(t)} = 1$. For all j, $1 \leq j \leq K$, the quality values are equal

$t \leftarrow 0$;
$(\mathbf{P}^{(0)}, \mathbf{Q}^{(0)}) \leftarrow$ InitializeAdaptationVectors ;
$s' \leftarrow$ GenerateInitialSolution ;
$s^{(0)} \leftarrow$ LocalSearch$(s', (\mathcal{N}_1, \ldots, \mathcal{N}_P))$;
while *The stopping criteria is NOT met* **do**
  $(v_{\mathcal{M}}, v_{\mathcal{N}}) \leftarrow$ SelectParameters$(\mathbf{P}^{(t)})$;
  $s' \leftarrow$ Perturbation$(s^{(t)}, \mathcal{M}, v_{\mathcal{M}})$;
  $\mathcal{N}^{(t)} \leftarrow$ SelectNeighbourhood$(\mathcal{N}, v_{\mathcal{N}})$;
  $s^{(t+1)} \leftarrow$ LocalSearch$(s^{(t)}, \mathcal{N}^{(t)})$;
  $\mathbf{R}_{\mathbf{v}}^{(t)} \leftarrow$ UpdateRewardVector$(v_{\mathcal{M}}, v_{\mathcal{N}})$;
  Update $\mathbf{Q}^{(t+1)}$ with reward $\mathbf{R}_{\mathbf{v}}^{(t)}$;
  $\mathbf{r} \leftarrow$ HighRankQ$(\mathbf{Q}^{(t+1)})$;
  $\mathbf{P}^{(t+1)} \leftarrow$ UpdateProbability$(\mathbf{P}^{(t)}, \mathcal{D}_{\mathbf{r}}, \beta)$;
  $t \leftarrow t + 1$
end
return *The local optimum solution* $s^{(t)}$

Algorithm 5: Adaptive Multi - operator MetaHeuristics (AMMH)

$\mathcal{Q}_{1j}^{(t)} \leftarrow 0.5$ meaning that all the parameters are considered equally important at the initialization. In the sequel, for the neighbourhood operator, we have $\mathcal{P}_{2i}^{(t)} \leftarrow 1/P$ and $\mathcal{Q}_{2i}^{(t)} \leftarrow 0.5$, where $\forall i, 1 \leq i \leq P$.

SelectParameters independently selects parameters for each operator. For the mutation operator, an operator $v_{\mathcal{M}} \in \mathcal{M}$ is selected proportionally with the corresponding probability distribution $\mathbf{P}_{\mathcal{M}}^{(t)} \leftarrow (\mathcal{P}_{11}^{(t)}, \ldots, \mathcal{P}_{1K}^{(t)})$. For the neighbourhood operator, an operator $v_{\mathcal{N}} \in \mathcal{N}$ is selected proportionally with the probability distribution $\mathbf{P}_{\mathcal{N}}^{(t)} \leftarrow (\mathcal{P}_{21}^{(t)}, \ldots, \mathcal{P}_{2P}^{(t)})$. The resulting parameter vector is denoted with $\mathbf{v} \leftarrow (v_{\mathcal{M}}, v_{\mathcal{N}})$.

UpdateRewardVector. An improvement in the cost of the candidate solution $s^{(t+1)}$ when compared with the cost of the current solution $s^{(t)}$ means that $c(s^{(t+1)}) < c(s^{(t)})$. The reward $\mathcal{R}_{\mathbf{v}}^{(t)}$ for using the vector of parameters $\mathbf{v}$ in restarting the local search is connected with the improvement over the current solution

$$\mathcal{Q}_{\mathcal{M}}^{(t)} = \frac{\#\ improv\ of\ v_{\mathcal{M}}}{\#\ trials\ of\ v_{\mathcal{M}}}, \qquad \mathcal{Q}_{\mathcal{N}}^{(t)} = \frac{\#\ improv\ of\ v_{\mathcal{N}}}{\#\ trials\ of\ v_{\mathcal{N}}}$$

In the initialization step, we ensure that all the operators are tried at least once. If applying $v_{\mathcal{M}}$ results in an improvement, $\mathcal{Q}_{\mathcal{M}}^{(t)}$ is increasing

$$\mathcal{Q}_{\mathcal{M}}^{(t)} < \mathcal{Q}_{\mathcal{M}}^{(t+1)} \Leftrightarrow \frac{\#\ improv\ v_{\mathcal{M}}}{\#\ total\ v_{\mathcal{M}}} < \frac{\#\ improv\ v_{\mathcal{M}} + 1}{\#\ total\ v_{\mathcal{M}} + 1}$$

Otherwise, $\mathcal{Q}_{\mathcal{M}}^{(t)}$ decreases

$$\mathcal{Q}_{\mathcal{M}}^{(t)} > \mathcal{Q}_{\mathcal{M}}^{(t+1)} \Leftrightarrow \frac{\#\ improv\ v_{\mathcal{M}}}{\#\ total\ v_{\mathcal{M}}} > \frac{\#\ improv\ v_{\mathcal{M}}}{\#\ total\ v_{\mathcal{M}} + 1}$$

Similar properties we have for $\mathcal{Q}_{\mathcal{N}}^{(t)}$.

Rank quality vectors. The function call HighRankQ independently high ranks the quality vectors for each operator $\mathbf{Q}_{\mathcal{M}}^{(t+1)}$ and $\mathbf{Q}_{\mathcal{N}}^{(t+1)}$. There is a quality ranking matrix $\mathbf{r}$ that has an independent quality vector for each operator, $\mathbf{r} = (\mathbf{r}_{\mathcal{M}}, \mathbf{r}_{\mathcal{N}})$. Thus, an instance of a quality vector $\mathbf{r}$ can be highly ranked for one operator and lower ranked for the other operator.

Update probabilities. The two probability distributions of $\mathbf{P}^{(t)}$ are also independently updated for each operator. For all j, $1 \leq j \leq K$, we have that

$$\mathcal{P}_{1j}^{(t+1)} \leftarrow \mathcal{P}_{1j}^{(t)} + \beta \cdot (\mathcal{D}_{r_{1j}} - \mathcal{P}_{1j}^{(t)})$$

and for all i, $1 \leq i \leq P$, we have that

$$\mathcal{P}_{2i}^{(t+1)} \leftarrow \mathcal{P}_{2i}^{(t)} + \beta \cdot (\mathcal{D}_{r_{2i}} - \mathcal{P}_{2i}^{(t)})$$

The target distribution, $\mathbf{D}$, is a step-like distribution for each operator. Thus,

$$\mathcal{D}_{\mathcal{M}} = [p_M, \underbrace{p_m, \ldots, p_m}_{K-1}], \text{ where } p_M = 1 - (K-1) * p_m$$

and

$$\mathcal{D}_{\mathcal{N}} = [p_N, \underbrace{p_n, \ldots, p_n}_{P-1}], \text{ where } p_N = 1 - (P-1) * p_n$$

For each operator, only one element has the maximum value p_M or p_N and the rest of the operators are updated with a low probability, p_m or p_n, after case. In updating the j-th element in $\mathcal{P}_{\mathcal{M}}^{(t)}$, the rank j in $\mathcal{D}_{\mathcal{M}}$ is used. If $\mathcal{D}_{\mathcal{M}}$ is a valid probability vector, i.e., $\sum_{j=1}^{K} \mathcal{D}_{1j} = 1$, then the elements in the probability vector $\mathcal{P}_{\mathcal{M}}^{(t)}$ sum up to 1. In the sequel, if $\mathcal{D}_{\mathcal{N}}$ is a valid probability vector, i.e., $\sum_{i=1}^{P} \mathcal{D}_{2i} = 1$, then the elements in the probability vector $\mathcal{P}_{\mathcal{N}}^{(t)}$ sum up to 1.

The algorithm. The adaptive multi-operator metaheuristics algorithm starts by initializing the adaptation probability distributions. The initial solution s' and the initial neighbourhood $\mathcal{N}^{(0)}$ are used to restart local search.

Each iteration, two parameters, $v_{\mathcal{M}}$ and $v_{\mathcal{N}}$, which is one parameter for each operator, are generated using the function SelectParameters. A new solution s' is generated using the perturbator function over the current solution $s^{(t)}$, and the given mutation parameter $v_{\mathcal{M}}$. A neighbourhood function $\mathcal{N}^{(t)}$ is selected from the set of neighbourhoods $\mathcal{N}$ as indicated by the parameter $v_{\mathcal{N}}$. The local search is restarted from solution s' using the neighbourhood $\mathcal{N}^{(t)}$. The reward vector is updated

Remark 4.1. Adaptive pursuit has two extra parameters that could be tuned: i) minimum (and maximum) selection probabilities, and ii) the learning rate β. Let $\beta \in [0, 1]$ be the learning rate that determines the speed with which the algorithm converges to the maximum and minimum estimated reward values. A large β value means faster convergence of the algorithm to the target probabilities, which means a poorer use of certain

operators. A small β value means slower convergence and thus more chances for the less rewarded operators to be tested.

Remark 4.2. From the three proposed multi-operator metaheuristics, Algorithm 5 and Algorithm 3 resemble the most since both algorithms will simultaneously use both operators to restart the local search, whereas Algorithm 4 alternatively uses the two operators.

5 Experimental results

In this section, we show preliminary experimental results that compare the performance of seven metaheuristics on two QAP instances.

Tested metaheuristics algorithms. We compare the performance of the following metaheuristics:

MLS the multi-restarted local search algorithm uniform randomly restarts LS;
ILS the iterated local search algorithm uses a set of mutation exchange rates $\mathcal{M}$;
ALS the adaptive iterated local search algorithm uses adaptive pursuit on a set of mutation exchange rates $\mathcal{M}$;
VNS the variable neighbourhood search algorithm uses a set of neighbourhood functions $\mathcal{N}$;
MMH the multi-operator metaheuristic algorithm uses simultaneously a set of mutation exchange rates $\mathcal{M}$ and a set the neighbourhood functions $\mathcal{N}$;
AMH the alternating operators metaheuristic algorithm alternates the restarting operators;
AMMH the adaptive multi-operator metaheuristic algorithm adaptively selects the parameters for mutation and the neighbourhood function.

Note that four algorithms can be classified as iterated local search algorithms, *MLS*, *ILS*, *ALS* and *AMMH*. Four of the above algorithms can be classified as variable neighbourhood search algorithms, *VNS*, *MMH*, *AMH* and *AMMH*. The *AMMH* algorithm can be classified both as ILS and VNS since it uses both type operators to restart LS.

Tested QAP instances. We use composite QAP (cQAP) instances (Drugan, 2014) to test the seven metaheuristic algorithms. These QAP instances are automatically generated such that they have the following properties: i) large size, ii) difficult and interesting to solve with both heuristics and exact algorithms, iii) known optimum solution, and iv) not trivial asymptotic behaviour (i.e., the difference between the lower and the upper cost functions does not go to 0 when the number of facilities goes to ∞).

For our experiments, we consider two cQAP instances with a medium number of facilities, $N = \{20, 32\}$. The optimum solution is always the identity permutation.

Setting the parameters. The set of the mutation exchange rates is set to $\mathcal{M} = \{3, 4, 5, 6, 7\}$. Thus, $K = |\mathcal{M}| = 5$. The set of neighbourhood functions cannot be very large for the QAP instances because the exploration of their neighbourhood increases quadratically with the exchange rates $\mathcal{N} = \{2, 3, 4\}$. Thus, $P = |\mathcal{N}| = 3$.

Each metaheuristic algorithm is run for 100 times for an equal amount of exchanges 10^6.

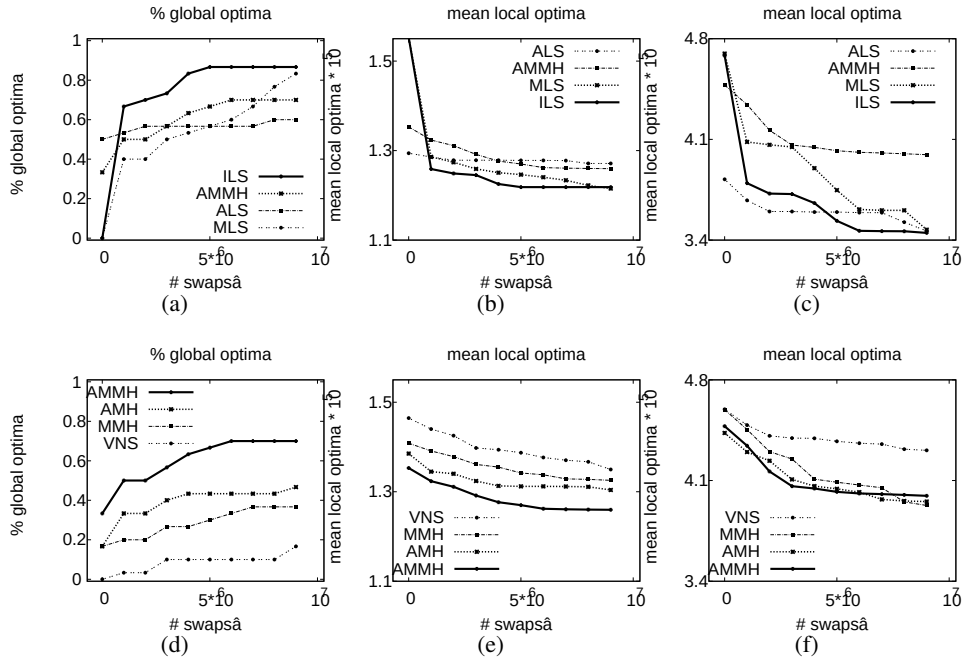

Fig. 1: (On the left) The percent of runs in which of the local optima value for cQAP with $N = 20$ is found for the seven algorithms. (In the middle) The mean cost function of the seven algorithms when $N = 20$, and (on the right) the mean cost function when $N = 32$. (On the top) The four iterative based metaheuristic algorithms, and (on the bottom) the four variable neighbourhood search based metaheuristic algorithms.

For the multi-operator adaptive pursuit, we consider a standard learning rate of $\beta = 0.01$. We set the maximum and the minimum probability distribution such that the proportion between these probabilities is 6. Then, for the mutation operator we have $p_M = 0.6$ and $p_m = 0.1$, and for the neighbourhood operator we have $p_N = 0.7$ and $p_n = 0.15$.

5.1 Results

In Figure 1, to compare the performance of the discussed metaheuristics, we measure the percentage of times the global optima is found. For $N = 32$, the number of identified local optima is 0 for all the algorithms showing the increase in complexity of the problem with the increase in number of facilities.

$N = 20$**.** The best algorithm is the iterated local search (*ILS*) followed by multi-restart local search (*MLS*). The worst algorithm is the variable neighbourhood search algorithm (*VNS*). In general, the meta-heuristics algorithms that use mutation perturbators, in Figure 1 on the left, outperform the algorithms that alternates the neighbourhood functions, in Figure 1 on the right. The adaptive multi-operator metaheuristics (*AMMH*)

has a better performance than both the multi-operator metaheuristics (*MMH*) and the adaptive iterated local search (*ALS*).

$N = 32$**.** For larger cQAP instances, the adaptive iterated local search (*ALS*) is performing similarly with the iterative local search (*ILS*). The adaptive multi-operator metaheuristics (*AMMH*) is again the best algorithm that uses variable neighbourhood search, but this time is outperformed by the adaptive iterated local search (*ALS*).

Discussion. We conclude that for smaller size cQAPs, adaptation of the neighbourhood function is more beneficial than the adaptation of the mutation operators, whereas for larger size cQAPs, this situation is vice-versa. This could indicate that the the usage of multi restarting operators can be beneficial in certain cases.

6 Conclusions

We have proposed a new class of metaheuristics that uses two operators to restart the search, i.e. the stochastic perturbator operators and the neighbourhood function. We motivate the multi-operator metaheuristic algorithms with the quadratic assignment problem where the size of the neighbourhood increases quadratically with the exchange rate, whereas the generation of a solution with mutation increases only linearly.

We propose two instances of multi-operator metaheuristic algorithms. One instance uses the two restarting techniques simultaneously, but the alternation of the neighbourhood functions is the most important procedure in the local search call. The second multi-operator metaheuristics algorithm uses the two operators alternatively: i) the mutation operator is used with the small neighbourhood function, and ii) the local search is now using a single, uniformly selected, neighbourhood function. We consider the last algorithm to be more balanced in using the two restarting operators than the first algorithm. The third proposed algorithm is an adaptive multi-operator metaheuristic algorithm where the adaptive pursuit is extended to a multi-objective adaptive pursuit. In this variant of multi-operator adaptive pursuit, both operators are adapted separately.

Preliminary experimental results compare the performance of the discussed metaheuristics and show a better performance of the multi-operator metaheuristics as compared with the variable neighbourhood search metaheuristics for the tested QAP instances. An interesting perspective is to apply the same methodology to solve multi-objective combinatorial optimization problems.

Bibliography

[Drugan, 2014]Drugan, M. M. (2014). Generating QAP instances with known optimum solution and additively decomposable cost function. *Jornal of Combinatorial Optimization*, to appear.

[Drugan and Thierens, 2010]Drugan, M. M. and Thierens, D. (2010). Path-guided mutation for stochastic Pareto local search algorithms. In *Proc of Parallel problem solving from Nature (PPSN'10)*, pages 485–495.

[Drugan and Thierens, 2011]Drugan, M. M. and Thierens, D. (2011). Generalized adaptive pursuit algorithm for genetic Pareto local search algorithms. In *Proc of Conf. of Genetic and Evol. Comp., (GECCO'11)*, pages 1963–1970.

[Drugan and Thierens, 2012]Drugan, M. M. and Thierens, D. (2012). Stochastic Pareto local search: Pareto neighbourhood exploration and perturbation strategies. *J. Heuristics*, 18(5):727–766.

[Hansen and Mladenovic, 2006]Hansen, P. and Mladenovic, N. (2006). First vs. best improvement: An empirical study. *Discrete Applied Mathematics*, 154(5):802 – 817.

[Hansen et al., 2008]Hansen, P., Mladenovic, N., and Perez, J. A. M. (2008). Variable neighbourhood search: methods and applications. *4OR*, 6(4):319–360.

[Hoos and Stutzle, 2005]Hoos, H. H. and Stutzle, T. (2005). *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann.

[Loiola et al., 2007]Loiola, E., de Abreu, N., Boaventura-Netto, P., Hahn, P., and Querido, T. (2007). A survey for the quadratic assignment problem. *European J. of Operational Research*, 176(2):657–690.

[Mladenovic and Hansen, 1997]Mladenovic, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097 – 1100.

[Stützle, 2006]Stützle, T. (2006). Iterated local search for the quadratic assignment problem. *European Journal of Operational Research*, 174(3):1519–1539.

[Talbi, 2009]Talbi, E.-G. (2009). *Metaheuristics: from design to implementation*. Wiley.

[Thierens, 2005]Thierens, D. (2005). An adaptive pursuit strategy for allocating operator probabilities. In *Proc of Conf. of Genetic and Evol. Comp., (GECCO'05)*, pages 1539–1546.