

Generalized Adaptive Pursuit Algorithm for Genetic Pareto Local Search Algorithms

Mădălina M. Drugan
Utrecht University
Department of Information and Computing
Sciences
PO Box 80.089, 3508TB Utrecht
The Netherlands
madalina@cs.uu.nl

Dirk Thierens
Utrecht University
Department of Information and Computing
Sciences
PO Box 80.089, 3508TB Utrecht
The Netherlands
dirk@cs.uu.nl

ABSTRACT

The standard *adaptive pursuit* technique (AP) shows preference for a single operator at a time but is not able to simultaneously pursue multiple operators. We generalize AP by allowing any target distribution to be pursued for operator selection probabilities. We call this the *generalized adaptive pursuit algorithm* (GAPA). We show that the probability matching and multi-armed bandit strategies, with particular settings, can be integrated in the GAPA framework. We propose and experimentally test two instances of GAPA. Assuming that there are multiple useful operators, the *multi-operator AP* pursues them all simultaneously. The *multi-layer AP* is intended to scale up the pursuit algorithm to a large set of operators. To experimentally test the proposed GAPA instances, we introduce the *adaptive genetic Pareto local search* (aGPLS) that selects online genetic operators to restart the Pareto local search. We show on a *bi-objective Quadratic assignment problem* (bQAP) instance with a large number of facilities and high correlation that aGPLSs are the algorithms with best performance tested.

Categories and Subject Descriptors: I.2.8 [Problem Solving, Control Methods, and Search]: Heuristic methods

General Terms: Algorithms

Keywords: Adaptive pursuit algorithm, Genetic Pareto local search

1. INTRODUCTION

Pursuit allocation strategies (AP) [5] are online operator selection algorithms that adapt a selection probability distribution such that the operator with the maximal estimated reward is pursued. The operator selection target distribution does not change in time. For AP, this 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.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GECCO '11, July 12–16, 2011, Dublin, Ireland.

Copyright 2011 ACM 978-1-4503-0557-0/11/07 ...\$10.00.

In this paper, we generalize the adaptive pursuit technique such that the target distribution of the operator selection probabilities can be *any* discrete distribution. We call this class of algorithms *generalized adaptive pursuit algorithms* (GAPA). We show that other popular techniques for online operator selection, probability matching and the multi-armed bandit strategies [1], can be thought of as instances of GAPA. We propose several new instances for GAPA. The *multi-operator AP* pursues multiple, ranked, preferences modeling a large range of target distributions. The *multi-layer AP* algorithm is a method to design GAPA with desired properties by using multiple layers of AP. These layers add flexibility to GAPA because the contained APs can have different target distributions and learning rates. We give examples of multi-layer AP that use fixed and dynamical tree structures to model the relationship between GAPAs with a smaller number of operators to scale up.

In the empirical study, we design adaptive operator selection algorithms for genetic Pareto local search (GPLS). Pareto local search (PLS) [4] iteratively moves from a current solution to a neighboring solution using an exploration strategy. The algorithm stops when no improving solution can be found in the neighborhood. Genetic PLS [2] restarts the PLS from promising regions of the search space with genetic operators like mutation and recombination operators. However, it has been pointed out [2] that its experimental performance is sensitive to the choice of the operators. We call this class of adaptive algorithms *adaptive GPLS* (aGPLS). We test these proposed algorithms on an instance of the *bi-objective quadratic assignment problem* (bQAP). QAPs are NP-hard combinatorial optimization problems that model many real-world problems (i.e., scheduling, vehicle routing).

Section 2 proposes the generalized adaptive pursuit algorithm (GAPA). Section 3 proposes instances of GAPA. Section 4 introduces the (adaptive) genetic Pareto local search (GPLS). Section 5 presents some experimental results. Section 6 concludes the paper.

2. GENERALIZED ADAPTIVE PURSUIT ALGORITHM (GAPA)

Algorithm 1 introduces the generalized adaptive pursuit algorithm (GAPA). It generalizes the standard adaptive pursuit (AP) algorithm [5] by allowing any discrete target distribution, \mathcal{D} , for the operator selection probabilities. Consider a set of k operators $\mathcal{A} = \{a_1, \dots, a_k\}$, a probability vector

Algorithm 1: GAPA($k, \mathcal{D}, \mathcal{R}, \beta, w$)

```
1: for i = 1 to k do
2:   t ← 0    $\mathcal{P}_i^{(t)} \leftarrow 1/k$ ;    $Q_i^{(t)} \leftarrow 0.5$ 
3: end for
4: while Stopping criteria  $\mathcal{T}$  is NOT met do
5:   Select an operator  $v$  using  $\mathcal{P}^{(t)}$ 
6:    $\mathcal{R}_v^{(t)} \leftarrow \text{GetRewardApplying}(v)$ 
7:   Call UpdateAdaptationVectors( $Q^{(t)}, \mathcal{P}^{(t)}$ )
8:   begin
9:     Update  $Q_v^{(t+1)}$  with reward  $\mathcal{R}_v^{(t)}$ 
10:    High-rank  $Q^{(t+1)}$  and set the indexes in  $r$ 
11:    for i = 1 to k do
12:       $\mathcal{P}_i^{(t+1)} \leftarrow \mathcal{P}_i^{(t)} + \beta \cdot (\mathcal{D}_{r[i]} - \mathcal{P}_i^{(t)})$ 
13:    end for
14:  end
15: end while
```

$\mathcal{P}^{(t)} = \{\mathcal{P}_1^{(t)}, \dots, \mathcal{P}_k^{(t)}\}$ ($\forall t: 0 \leq \mathcal{P}_i^{(t)} \leq 1; \sum_{i=1}^k \mathcal{P}_i^{(t)} = 1$), and the quality vector (or estimated reward) $Q^{(t)}$.

First, an operator v is selected from the distribution $\mathcal{P}^{(t)}$. Applying this operator means that a new solution is generated using it and PLS is restarted from this solution. A reward $\mathcal{R}_v^{(t)}$ is returned after applying the operator v . Each measurement of performance for v has a time stamp, t' . Between lines 7 and 13 in Algorithm 1 is the body of procedure *UpdateAdaptationVectors* which updates the vectors $\mathcal{P}^{(t)}$ and $Q^{(t)}$. The update algorithm for the quality vector, $Q^{(t)}$, and the reward $\mathcal{R}_v^{(t)}$ depend on the type of problem used, here multi-objective QAPs, and is presented in Section 4.

The update of $\mathcal{P}^{(t)}$ is started by ranking the quality vector $Q^{(t)}$ at line 10. The rank vector, r , ranks the most rewarded operator with 1, the operator with maximum $Q^{(t)}$. In updating the i -th element in $\mathcal{P}^{(t)}$, the rank i in \mathcal{D} is used. If \mathcal{D} is a valid probability vector, i.e., $\sum_{i=1}^k \mathcal{D}_i = 1$, then the elements in the probability vector $\mathcal{P}^{(t)}$ sum up to 1. 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.

2.1 Adaptive selection strategies in GAPA

We consider three different types of online operator selection methods: i) probability matching, ii) adaptive pursuit, and iii) multi-armed bandit. We demonstrate that these different adaptive operator selection strategies fit in the GAPA framework. In Algorithm 1, the only condition that \mathcal{D} needs to fulfill is that it is a valid distribution with probabilities summed up to 1.

Standard adaptive pursuit. The classical pursuit algorithm [5] adapts the probability vector \mathcal{P}^t such that the operator that has the highest estimated reward is chosen with very high probability, p_M . Like GAPA, AP has two parameters: i) minimum (and maximum) selection probabilities, and ii) the learning rate β . The target distribution, \mathcal{D} , is then a step-like distribution where only one ele-

ment has the value p_M and the rest $k - 1$ of the operators are updated with a low probability, p_m . In Algorithm 1, $\mathcal{D} = [p_M, p_m, \dots, p_m]$, where $p_M = 1 - (k - 1) * p_m$.

Probability matching in GAPA. *Probability matching* selects an operator with a probability proportional with its estimated reward (or quality) function. We now show that the framework from Algorithm 1 is general enough to include also the *probability matching* adaptation technique. Let the target distribution \mathcal{D} be defined by the quality vector, $Q^{(t)}$, normalized over all operators v . Then, $\mathcal{D}_v^{(t+1)} = Q_v^{(t+1)} / \sum_{i=1}^k Q_i^{(t+1)}$. In Algorithm 1, if $\beta = 1$, then $\mathcal{P}_i^{(t+1)} = \mathcal{D}_{r[i]}^{(t+1)}$. Here, $Q^{(t)}$ and $\mathcal{P}^{(t)}$'s elements have the same ranking meaning that $r[i] = i, \forall i$. We conclude that for this \mathcal{D} depending on the reward function and for $\beta = 1$, Algorithm 1 represents the *probability matching* adaptation technique. There are variants of probability matching [5] that set minimum and maximum selection probabilities, a concept that integrates well in our framework.

Note that the biggest differences between \mathcal{D} for adaptive pursuit and probability matching are that, for the first, the target distribution is a priori selected, stationary and independent of the quality vector, $Q^{(t)}$, and, for the later, \mathcal{D} is dependent of time and $Q^{(t)}$.

Multi-armed bandit in GAPA. Another important class of online adaptive selection algorithms are the bandit based adaptive selection algorithms [1] initially used for multi-armed bandit problems. In the multi-armed bandit framework (MAB), each operator is considered an arm with an unknown probability of getting a reward. Each arm v is associated with: i) a quality vector, $Q_v^{(t)}$, and ii) a confidence interval $C \cdot \sqrt{2 \cdot \ln \sum_v \# total v / \# total v}$, where $\# total v$ is the number of times the v -th arm is tried and $\sum_v \# total v$ is the total number of times all operators are tried, and C a real valued parameter determined experimentally. For large C , $Q^{(t)}$ is dominated by the confidence interval whereas for small C , there is very little exploration and the adaptation is stuck for a long time.

In Algorithm 1, if $\beta = 1$, then $\mathcal{P}_i^{(t+1)} = \mathcal{D}_{r[i]}^{(t+1)}$. For MAB, we have $\mathcal{D}_{r[1]}^{(t+1)} = 1$, and $\mathcal{D}_{r[i]}^{(t+1)} = 0, \forall i > 1$, where $Q^{(t)}$ and $\mathcal{P}^{(t)}$'s elements have the same ranking meaning that $r[i] = i, \forall i$. Each time step, the arm v which maximizes the estimated reward is chosen. Like the standard AP, MAB is focused on selecting only the best performing operator. Unlike MAB, AP has non-zero selection probabilities for the other operators and the learning rate is $\beta \in [0, 1]$.

We conclude that GAPA generalizes the three previously introduced strategies. Unlike standard AP and MAB, GAPA does not focus on a single, best, operator according with some reward related function. It also generalizes the probability matching because, in GAPA, the target distribution is independent of actually obtained rewards.

3. NEW INSTANCES OF GAPA

To design meaningful GAPA instances we need a systematic technique to generate target distributions with desired properties. In this section, we design two types of instances of GAPA that can pursuit more than one operator. We assume that exploiting a set of related operators instead of just one operator can be beneficial for the performance of the adaptive algorithm. The next instances can be combined with the existent adaptive strategies, which can be consid-

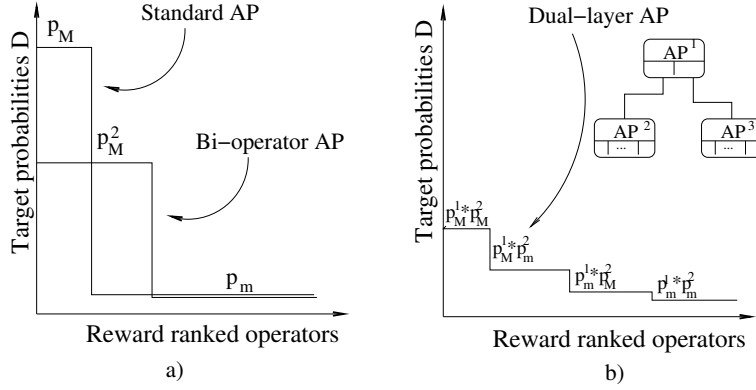


Figure 1: Target distributions \mathcal{D} for a) the standard AP and the bi-operator AP, and b) the dual-layer AP.

ered special instances of GAPA, generating new adaptation algorithms.

3.1 Multi-operator adaptive pursuit

We propose two multi-operator adaptive pursuit techniques that assume that a set of operators improve the performance of the algorithm more than a single operator does. The first technique pursuits with equal probability a fixed number of operators. The second technique assumes that different operators are useful in a different degree for the adaptation algorithm. The operator selection target distribution, for example, is a power distribution.

k-operator adaptive pursuit. It uses a straightforward generalization of the step distribution for \mathcal{D} where an a priori known number of operators, $k_M \geq 1$, are set to the maximum value p_M . The rest of the operators, $k - k_M$, are set to the minimum values p_m . In order for $\mathcal{D} = [p_M, \dots, p_M, p_m, \dots, p_m]$ to be a valid distribution, the following should hold

$$p_M = [1 - (k - k_M) * p_m] / k_M$$

Let's assume that p_m , where $p_m < p_M$, is the same for any value of $k_M \in \{1, \dots, k - 1\}$. Then, the value of p_M is decreasing when k_M is increasing. The difference in values between $p_M^{k_M}$ and the maximum value from the standard AP, p_M^1 , is

$$p_M^{k_M} = \frac{1 - (k - k_M) * p_m}{k_M} = \frac{p_M^1}{k_M} + (1 - \frac{1}{k_M}) * p_m$$

where $p_M^1 = 1 - (k - 1) * p_m$. For large values of k_M and small values for p_m , we have $p_M^{k_M} \approx p_M^1 / k_M$. This means that the best operator selected with the standard AP is selected k_M times more than when k_M operators are pursuit in the same time. On the other hand, $k_M - 1$ operators that were pursuit with probability p_m , are now pursuit with probability $p_M \gg p_m$. In Figure 1 a), we sketch the differences between the standard AP algorithm and a multi-operator AP which pursuits two operators, called *bi-operator AP*.

Power distribution adaptive pursuit. Let's assume a power 2 target distribution for adaptive pursuit and k such that $\log_2 k$ is an integer. The target distribution is $D = [p_M, p_0, p_1, p_1, \dots, p_m]$, where 2^i operators have the selection probability of p_i , $i = 0, \dots, \log_2 k - 1$. We consider that $p_m = p_{\log_2 k - 1} \leq p_{\log_2 k - 2} \leq \dots \leq p_0 \leq p_M$. \mathcal{D} is a distribution if $p_M + \sum_{j=0}^{i-1} 2^j \cdot p_j = 1$. In Algo-

rithm 1, for the operator with the largest estimated reward, $r[1]$, $\mathcal{P}^{(t)}$ is updated with the maximum selection probability $D_{r[1]} = p_M$. Half, $k/2$, of the operators with the lowest estimates, $\{r[k/2 + 1], \dots, r[k]\}$, updates $\mathcal{P}^{(t)}$ using the minimum probability p_m . The rest of $k/2 - 1$ operators, $\{r[2], \dots, r[k/2]\}$, are updated using probability values between p_m and p_M , such that the higher the rank of the operator the higher the probability to be selected.

3.2 Multi-layer adaptive pursuit

Assume now that there are sets of operators that are performing better. For example, the experimental results in [2] show a different dynamics, and performance, for mutation and recombination like operators for bQAPs. A simple or multi-operator AP cannot learn how good sets of operators are. Multi-layer AP uses multiple layers of AP, each with different target distributions and learning rates, in order to exploit sets of operators. The example is a simple dual-layer AP which is experimentally tested on bQAPs. Dual-layer AP has a fixed structure where all the operators are belonging to a fixed (set of) AP units. As for bi-operator AP, we make assumptions on how many operators are useful. The bi-operator AP assumes that the operators with the maximum selection probabilities can be any of the k operators. *Dual-layer AP* assumes that the distribution of good operators in sets of operators can be specified and a biased selection of one of the sets is beneficial. We also sketch a multi-layer AP, *dynamical structure multi-layer AP*, that uses a dynamical policy to assign operators to a layer and which does not make a priori assumptions about the relationships between operators.

Dual-layer adaptive pursuit. We assume that there are sets of operators, grouped in an AP, that improve the performance of the adaptation algorithm better than other sets of operators. Let's consider a simple tree structure with a parent and two children. Each of these nodes are AP units. The parent AP belongs to layer 1, we call it AP^1 , and its children belong to layer 2, AP^2 and AP^3 . AP^1 has only two states $k^1 = 2$, $\mathcal{D}^1 = [p_M^1, p_m^1]$, and the selection probability vector is denoted with $\mathcal{P}^{1,(t)}$. In layer 2, half of the operators, $k^2 = k/2$, belongs to one of the children AP^2 , and the other half to the other child AP^3 , where the split was made based on a priori knowledge. Let's denote with $\mathcal{D}^2 = \mathcal{D}^3 = [p_M^2, \dots, p_M^2, p_m^2, \dots, p_m^2]$ the target distribution of AP^2 and AP^3 , and with $\mathcal{P}^{2,(t)}$ and $\mathcal{P}^{3,(t)}$ the cor-

responding selection probability vectors. Figure 1 b) shows the structure of a dual-layer AP.

Operator selection with $\mathcal{P}^{(t)}$. To select an operator v , line 5 in Algorithm 1, the nodes in the tree are visited top-down. First, AP^1 selects one of its two states using the probability vector $\mathcal{P}^{1,(t)}$. AP^2 or AP^3 that corresponds to the selected state in AP^1 is further selected and an operator v is chosen from it using $\mathcal{P}^{2,(t)}$ or $\mathcal{P}^{3,(t)}$. The probability to select an operator v is now

$$\mathcal{P}_v^{(t)} = \begin{cases} \mathcal{P}_1^{1,(t)} \cdot \mathcal{P}_v^{2,(t)} & \text{if } 1 \leq v \leq k/2 \\ \mathcal{P}_2^{1,(t)} \cdot \mathcal{P}_{v-k/2}^{3,(t)} & \text{if } k/2 < v \leq k \end{cases}$$

For each of the two target distributions, \mathcal{D}^1 and \mathcal{D}^2 , there are two distinct values, p_M^1 and p_m^1 , and p_M^2 and p_m^2 , respectively. Therefore, as in Figure 1 b), there are four different selection probability values: i) $p_M^1 \cdot p_M^2$, ii) $p_m^1 \cdot p_M^2$, iii) $p_M^1 \cdot p_m^2$, and iv) $p_m^1 \cdot p_m^2$. The minimum probability to select an operator is now $p_m^1 \cdot p_m^2$ and has to be bounded such that the least rewarded operator is still fairly frequently selected. The maximum probability to select an operator is $p_M^1 \cdot p_M^2$ and should be chosen such that $p_M^1 \cdot p_M^2 \gg p_m^1 \cdot p_m^2$.

Updating $\mathcal{Q}^{(t)}$ and $\mathcal{P}^{(t)}$. The updating of $\mathcal{Q}^{(t)}$ is made from the leaves to the top of the tree. The reward is given, as before, to an individual operator v . If $v \leq k/2$, $\mathcal{Q}_v^{2,(t)}$ from AP^2 is updated. In layer 1, state 1 is updated with the same reward as $\mathcal{Q}_v^{2,(t)}$. Otherwise, if $k/2 < v \leq k$, then $\mathcal{Q}_{v-k/2}^{3,(t)}$ from AP^3 and $\mathcal{Q}_2^{1,(t)}$ from AP^1 are updated. The rest of the updates from Algorithm 1, lines 10-13, are made according with the variables in each AP and independently of other AP units in the tree.

Dynamical structure multi-layer AP. This strategy has $\log_2 k > 1$ number of layers, where k is a power of 2. Each layer $i = 1, \dots, \log_2 k$ is an adaptive pursuit with $k/2^{i-1}$ operators. In Algorithm 1, the update of $\mathcal{P}^{(t)}$ starts with AP^1 which has the largest number of operators, k . Half lowest ranked operators in AP^1 , $\{r[k/2+1], \dots, r[k]\}$, use the minimum selection value p_m^1 to update $\mathcal{P}^{1,(t)}$. The other half of operators are updated using the maximum operator selection probability p_M^1 , and afterwards are selected for the next layer, AP^2 . This updating process is iteratively repeated for each of the i layers, AP^i . Half lowest ranked operators in AP^i , $\{r[k/2^i+1], \dots, r[k/2^{i-1}]\}$ are updated with the minimum selection value for AP^i , p_m^i , and the other half with the maximum value for AP^i , p_M^i . The last layer has only 2 operators. The operator with the lowest rank is updated with $p_m^{\log_2 k}$ and the other operator with $p_M^{\log_2 k}$. The selection probability for an operator that is ranked high in $i-1$ layers and is low in AP^i , is $p_m^i \cdot \prod_{j=1}^{i-1} p_M^j$. The highest probability for an operator is $\prod_{j=1}^{\log_2 k-1} p_M^j$. Note the resemblance between this multi-layer AP and the power distribution AP.

Multi-layer AP in multi-operator AP. Even when the final target distribution of the two algorithms is the same, there are two large differences in their behavior: i) the use of learning rates, and ii) the distribution of the operators with maximum selection probabilities. The multi-operator AP uses a constant learning rate β , which is in general experimentally chosen. *Multi-layer* AP can assign different learning rates to different AP units in the tree. We assume that the layers that are first updated should converge slower to the target distribution because they learn to select an en-

Algorithm 2: GPLS($\mathcal{N}, \mathcal{T}, \mathcal{P}^{(t)}$)

```

1: Initialization:  $NDA_G \leftarrow \emptyset, t \leftarrow 0$ 
2: while Stopping criteria  $\mathcal{T}$  is NOT met do
3:   Generate sol  $s$  using an oper of  $\mathcal{P}^{(t)}$  // (5: GAPA)
4:   Call  $PLS(s)$ 
5:   begin
6:      $NDA \leftarrow \{s\}$ 
7:     while  $\exists s' \in NDA, s'.visited = false$  do
8:       for all  $s'' \in \mathcal{N}(s')$  do
9:         if  $s'' \notin NDA$  and  $\forall s''' \in NDA, f(s''') \preceq f(s'')$ 
           then
10:          Remove  $\forall s''' \in NDA, f(s''') < f(s'')$ 
11:          Add  $s''$  to  $NDA$ 
12:           $s''.visited \leftarrow false$ 
13:        end if
14:      end for
15:      Set  $s'.visited \leftarrow true$ 
16:    end while
17:     $\mathcal{R}_v^{(t)} \leftarrow \text{GetRewardApplying}(v)$  // (6: GAPA)
18:    UpdateAdaptationVectors( $\mathcal{Q}^{(t)}, \mathcal{P}^{(t)}$ ) // (7: GAPA)
19:    Merge  $NDA$  with  $NDA_G$ 
20:     $t \leftarrow t+1$  // (14: GAPA)
21:  end while
22: return  $NDA_G$ 

```

tire block of operators as best operators. Then, these layers are assigned lower β 's than layers updated afterwards.

4. GENETIC PARETO LOCAL SEARCH

Intuitively, a genetic PLS (GPLS) is a Pareto local search [4] that iteratively restarts the local search from solutions generated using a distribution specified by genetic operators. Consider a solution space \mathcal{S} , a countable ℓ -dimensional objective space \mathcal{O} , $\ell > 0$, and a function $f = (f_1, \dots, f_\ell) : \mathcal{S} \rightarrow \mathcal{O}$. Note that there are ℓ objectives and f_i is the fitness function in the i -th objective. We denote with $s \in \mathcal{S}$ a solution and $\mathcal{N}(s)$ a neighborhood of s . Consider the optimization problem where the order relationship \preceq associated with \mathcal{O} establishes a binary relationship between any two elements in \mathcal{O} . We say that a solution s *dominates* another solution s' , $f(s') < f(s)$, if $\exists i, f_i(s') < f_i(s)$ and $\forall j \neq i, f_j(s') < f_j(s)$ or $f_j(s') = f_j(s)$. We say that s' is incomparable with s if $\exists i, f_i(s') < f_i(s)$ and $\exists j, f_j(s) < f_j(s')$. We say that s is *not-dominated* by s' , $f(s') \preceq f(s)$, if: i) $f(s') < f(s)$, or ii) $f(s') = f(s)$, or iii) s' is incomparable with s . Because mQAP is a minimization problem, for an objective i , $<$ is equivalent with $>$.

The best set of non-dominated solutions, which dominates the other solutions outside this set, is called the *non-dominated archive* (NDA). For ease of discussion, we consider an unlimited size of the archive that can store *all* the solutions in the Pareto front. For very large fronts, i.e., real coded problems, where unlimited size NDAs are unrealistic, methods like ϵ -dominance can be applied.

Algorithm 2 presents the pseudocode of a genetic PLS algorithm. The NDA is initialized with a solution s that is generated from operators selected with a specified distribution $\mathcal{P}^{(t)}$, e.g. uniform random, or uniform randomly choosing from a set of operators, or pursuing (a subset of) operators. In PLS, each solution s has attached a flag, $s'.visited$ which

is set to *true* after evaluating the entire neighborhood of s' and *false* otherwise. Each iteration of PLS, a solution s' with the visited flag set to false, $s'.visited = false$, is randomly chosen from the current NDA. All the neighborhood solutions, $s'' \in \mathcal{N}(s')$, are evaluated. A solution s'' is added to NDA if it is not dominated by any solution in the current NDA. The solutions in NDA that are dominated by s'' are then removed from NDA. The search continues until there are no solutions left in the NDA that have their visiting flag set to false. Note that PLS is a best improvement algorithm because it selects all non-dominated neighboring solutions.

The PLS algorithm stops in a local optimal NDA. To find the best optima of the search space, GPLS restarts PLS multiple times and stops using a stopping criteria \mathcal{T} . Examples for \mathcal{T} can be the number of times a PLS is restarted or can depend on the type of problem GPLS is applied on.

GPLS for mQAPs. Intuitively, QAPs can be described as the (optimal) assignment of N facilities to N locations. A distance is specified between each pair of locations, and for each pair of facilities the amount of materials (or flow) transported between these facilities is given. The goal is to find the assignment of facilities to locations that minimizes the sum of the products between distances and flows.

We consider the multi-dimensional QAPs (mQAPs) introduced by Knowles and Corne [3]. These mQAPs have for each dimension different flow matrices and a single distance matrix. The flow matrices are correlated with some correlation ρ . Let's consider n facilities, a set $\Pi(n)$ of all permutations of $\{1, 2, \dots, n\}$ and the $n \times n$ distance matrix $A = (a_{ij})$, where a_{ij} is the distance between location i and location j . We assume an ℓ -dimensional space, and ℓ flow matrices $B^k = (b_{ij}^k)$, each with $n \times n$ elements, where b_{ij}^k represents the flow from facility i to facility j in the k -th objective dimension. The goal is to minimize for all dimensions the set of functions

$$f^k = C^k(\pi) = \sum_{i=1}^n \sum_{j=1}^n a_{ij} \cdot b_{\pi(i)\pi(j)}^k$$

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

QAPs are permutation problems, and a suitable neighborhood operator for PLS, \mathcal{N} , is the exchange operator that swaps the position of two or more facilities. For example, the 2-exchange swapping operator, swaps the position of two different facilities. The 2-exchange operator is attractive because of its linear time to compute the change in the cost function with the condition that all matrices A and B^k are symmetrical [4].

A GPLS for mQAPs uses two operators introduced in [2]. The path-guided mutation generates individual solutions on the path to another individual solution, whereas mutation generates solutions in any direction. The mutation and path-guided mutation respect cycles when swapping g pairs of positions. The parameter $g > 0$ has a large set of possible values: $3 \leq g \leq n$. In [2], we observed that the GPLS's performance greatly depends on the setting of g . The stopping criteria \mathcal{T} is chosen to fairly compare the performance of two GPLS algorithms. The distance between two solutions is defined as the minimum number of exchanges to obtain one solution from another solution. \mathcal{T} is then met after a fixed number of swaps.

Types of GPLS. The *multi-restart PLS (mPLS)* can be considered a GPLS that restarts the PLS from uniform ran-

domly generated solutions. mPLS was proposed by Paquete et al. [4]. There are certain limitations in mPLS design like it is basically random sampling in the space of local optima which inefficiently scales up to a large number of local optima.

To improve on multi-restart PLS's efficiency, *genetic PLS (GPLS)* [2] exploits the structure of the search space using genetic operators like mutation and recombination. We assume that the individuals generated with mutation or recombination from solutions in the current Pareto front are "good" solutions that are incomparable with solutions in the front. The choice of the operator and its parameter is specified by a fixed random distribution or by an adaptive distribution for adaptive GPLS.

4.1 Adaptive GPLS (aGPLS)

An aGPLS is a type of genetic PLS where the operator and its parameters that generates the restarting solution is chosen with an adaptive technique like the one in Algorithm 1. The adaptive strategy is repeatedly called after a PLS run until the stopping criteria \mathcal{T} from Algorithm 2 are met. To implement instances of adaptive GPLS, we first need to define the reward function, $\mathcal{R}_v^{(t)}$.

Updating $\mathcal{Q}_v^{(t)}$. The reward $\mathcal{R}_v^{(t)}$ for using the operator v in generating the restarting solution s is connected with the dominance relationship between the resulting archive, NDA_G , and the previous archive, NDA , from Algorithm 2.

$$\mathcal{Q}_v^{(t)} = \frac{\# \text{ improvements of } v \text{ operator}}{\# \text{ total trials of } v \text{ operator}} \quad (1)$$

where an improvement of using the operator v means that PLS is restarting from a solution generated with v and the resulting NDA contains at least one new solution, $s \in NDA$, that is dominating at least one other solution from the current archive NDA_G , $\exists s' \in NDA_G$ such that $f(s') \prec f(s)$. For the initialization step, to ensure that all the operators are tried at least once, we set $\mathcal{Q}_v^{(0)} = 0.5$, $\forall v \in \{1, \dots, k\}$, and $\# \text{ improv} = 1$ and $\# \text{ total } v = 2$. If applying v results in an improvement, $\mathcal{Q}_v^{(t)}$ is increasing

$$\mathcal{Q}_v^{(t)} < \mathcal{Q}_v^{(t+1)} \Leftrightarrow \frac{\# \text{ improv}}{\# \text{ total } v} < \frac{\# \text{ improv} + 1}{\# \text{ total } v + 1}$$

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

$$\mathcal{Q}_v^{(t)} > \mathcal{Q}_v^{(t+1)} \Leftrightarrow \frac{\# \text{ improv}}{\# \text{ total } v} > \frac{\# \text{ improv}}{\# \text{ total } v + 1}$$

For multi-armed bandit selection-like strategies the quality vector is

$$\mathcal{Q}_v^{(t)} = \frac{\# \text{ improv}}{\# \text{ total } v} + C \cdot \sqrt{\frac{2 \cdot \ln \sum_v \# \text{ total } v}{\# \text{ total } v}} \quad (2)$$

where C is a constant parameter as before.

The time window. To decrease the influence of old contributions of operators, in Algorithm 1, the measurements of performance that are older than w , with time stamp $t' < t - w$, are removed from $\mathcal{Q}_v^{(t)}$. This is necessary because of the non-stationarity of the environment: an operator that was very useful in the earlier stages of the search can be less useful in the later stages of the search and vice-versa. If the operator v is not tried out for a long time, then $\mathcal{Q}_v^{(t)}$'s value is 0.5, which is its initial value and the algorithm is urged to try out again v . Alternatively, we could use a learning

A. Hypervolume unary indicator.

<i>mPLS</i>	<i>rGPLS</i>	<i>aGPLS</i>	<i>gGPLS</i>	<i>tGPLS</i>	<i>aBGPLS</i>	<i>gBGPLS</i>	<i>tBGPLS</i>
0.51 ± 0.03	0.79 ± 0.09	0.80 ± 0.08	0.84 ± 0.07	0.83 ± 0.06	0.79 ± 0.09	0.82 ± 0.09	0.81 ± 0.08

B. Two sided non-parametrical Wilcoxon tests, column vs row.

<i>mPLS</i>	yes	yes	yes	yes	yes	yes	yes
<i>rGPLS</i>		no	yes	yes	no	no	no
<i>aGPLS</i>			yes	yes	no	no	no
<i>gGPLS</i>				no	yes	no	yes
<i>tGPLS</i>					yes	no	no
<i>aBGPLS</i>						no	no
<i>gBGPLS</i>							no

Table 1: Comparing the performance of the eight tested (a)GPLSs

rate, $\alpha \in [0, 1]$, for $Q_v^{(t)}$ as in [5] that exponentially lowers the weight of a past reward contribution.

5. EXPERIMENTAL RESULTS

In this section, we compare the behavior of two popular adaptation strategies summarized in Section 2 - that are adaptive pursuit and multi-armed bandit - with two of the adaptation strategies introduced in Section 3. Due to the lack of space, we do not show here empirical results for the power distribution and dynamical multi-layer GAPA. Instead, we have chosen to compare the already existent adaptation strategies that pursuit only *one* operator at the time with bi-operator and dual-layer adaptive pursuit algorithms that pursuit *two* operators. We also compare the adaptive strategies with uniform random choice of operators in order to establish if parameter adaptation is useful for these noisy environments like bi-QAPs. The popular multi-restart PLS algorithm is used as a base-line for comparisons with similar algorithms.

Although the principles discussed and the algorithms proposed in this paper are general, we limit our experiments to two objectives because it is easier to visualize the results of the algorithms.

The tested problem. We compare eight (a)GPLS algorithms on a bi-objective QAP (bQAP) instance generated using the software of Knowles and Corne [3]. This bQAP has high positive correlations $\rho = \{0.75\}$ and a large number of facilities $n = \{50\}$. To facilitate comparisons, we used an unstructured bQAP instance from Paquete’s study [4]¹. For bQAPs with a large number of facilities and high positive correlation Paquete reported a poor performance of multi-restart PLSs.

The tested algorithms. The performance of eight (a)GPLSs is tested. Six of them are adaptive GPLSs where three of them use the quality vector from Equation 1 and three of them have the multi-armed bandit-like quality vector from Equation 2. The multi-restart PLS, *mPLS*, with a best improvement 2-exchange neighborhood is restarted $M = 5000$ times. The number of swaps S is counted and that is the stopping criterion for the other six algorithms. We have used $S \approx 4 \cdot 10^8$. All the algorithms are run 50 times.

rGPLS is a *GPLS* where the restarts are generated uniform randomly with 32 mutation and 32 path-guided mu-

tation operators. The number of exchanges for both operators range in $g = \{3, \dots, 35\}$. Each operator in *rGPLS* is chosen with a probability $1.0/(32 + 32) \approx 0.015$. In our experiments, in a *rGPLS*’s run, PLS is restarted around 10^4 times, which means that all the operators are tried out, on average, $10^4/64 = 156$ times.

aGPLS is a *GPLS* where the distribution of operators for the restarting points are online learned with the standard AP, where operators have the same range as for *rGPLS*. Then, $k = 64$. We set $p_m = 0.01$, and then $p_M = 0.37$, and a low learning rate $\beta = 0.1$. Preliminary experiments with various β values showed variance in performance for the tested adaptive GPLS and a different optimal value for each algorithm. However, due to the limited space we show the performance for a single β value. We have chosen the time window $w = 2000$ which is about one fifth of 10^4 , the number of times PLS is restarted with *rPLS*.

gGPLS is an *aGPLS* which uses *bi-operator AP* as adaptive strategy. It has the same settings as *aGPLS* with exception $k_M = 2$ and $p_M = 0.19$.

tGPLS is an *aGPLS* which uses *dual-layer AP* to discriminate between the set of mutation operators and the set of path-guided mutation operators. The second layer of *tGPLS* contains an AP for the 32 mutation operators and an AP for the 32 path-guided mutation operators. For AP^2 and AP^3 in layer 2, we select $p_m^2 = 0.02$ and, then, we calculate $p_M^2 = 1 - 31 * p_m^2 = 0.38$. For AP^1 in layer 1, we select $p_m^1 = 0.34$ and, then, $p_M^1 = 0.66$. The target distribution of *tGPLS* takes one of the four values: 0.0068, 0.0132, 0.1292 and 0.2508. For the first layer, the learning rate is set to 0.05, and for AP^2 and AP^3 we set $\beta = 0.1$, which is the same learning rate as for the other adaptive GPLS, *aGPLS* and *gGPLS*.

aBGPLS, *gBGPLS* and *tBGPLS* are *aGPLS*, *gGPLS* and *tGPLS*, respectively, with a multi-armed update function from Equation 2. By construction, $p_M = 1.0$, $p_m = 0.0$ and $\beta = 1.0$. The size of the time window remains the same. To set C , we calculate the confidence interval for $w = 2000$, which is ≈ 0.78 and $\# total v = 31$, the number trials for an operator v if all the operators are randomly selected. Since the first part of Equation 2 belongs to the interval 0 and 1, we set C to a low value 0.1 such that the confidence interval has a lower weight. We have performed preliminary experiments with higher valued C (e.g., 0.5 and 1.0) and the performance of the algorithms decreases.

Performance measures. The *hypervolume* indicator is a unary performance measure designed to compare the Pareto fronts given by multi-objective combinatorial opti-

¹The bQAP instance used is <http://eden.dei.uc.pt/paquete/qap/instances/qapUni.50.p75.1.gz>. We also use the corresponding reference solutions provided on the same site and we refer to it as *NDA**

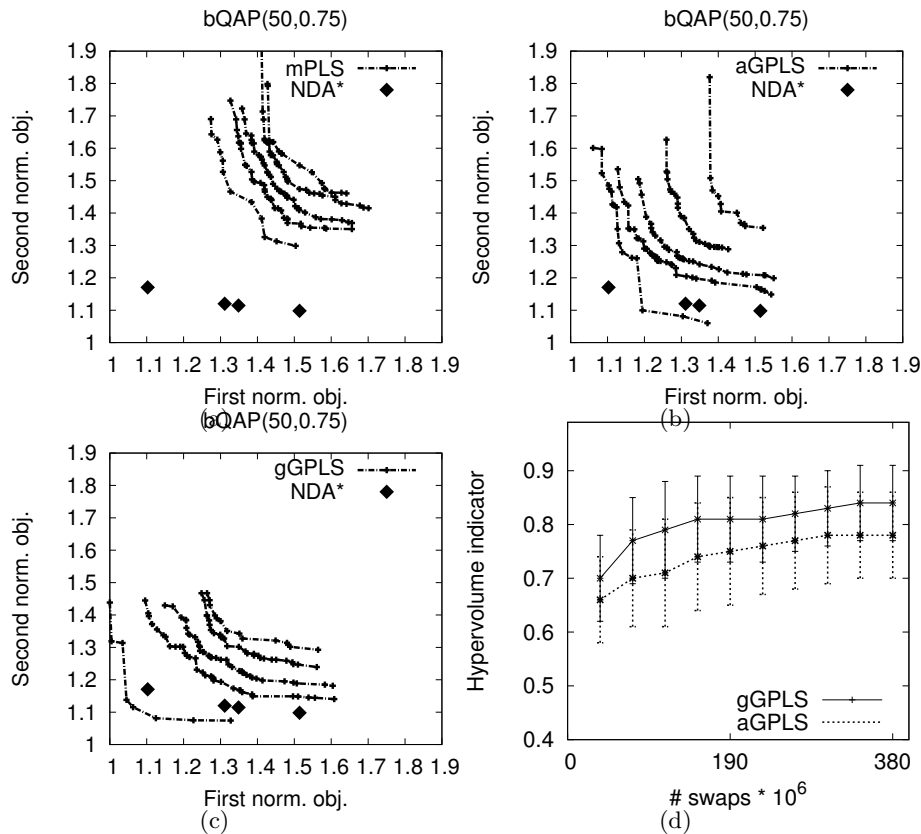


Figure 2: The attainment surfaces at 2%, 25%, 50%, 75% and 100% (lines from bottom left to top right) of the normalized outputs of a) *mPLS*, b) *aGPLS* and c) *gGPLS*. The objective values found at 2% EAFs correspond to the best NDA found over 50 runs and those found at 50% EAFs are the median outcome. d) The hypervolume indicators and the standard deviations of *aGPLS* and *gGPLS*.

mization algorithms [6]. The larger the hypervolume, the larger is the volume between a reference point, in this case the worst point in the bi-variate normalized search space, and the Pareto front. The larger the hypervolume the better the algorithm performs. Another method to compare the performance of two (and more) algorithms are the *attainment functions* (EAF). The unary *attainment function* gives the probability of attaining each point (independently) in the objective space. Certain contour surfaces through certain probabilities can then be drawn. For comparison purposes, a normalization function assigns to the best point(s) in a dimension the value 1 and to the worst point(s) the value 2. All the other points are scaled to a value between 1 and 2 in both dimensions, and the reference point is $\{2, 2\}$.

Results. Table 1 shows the hypervolume unary indicators for the eight tested algorithms and if their results are statistically significant different or not. In Figure 2 we show the EAFs for three of the algorithms tested to visually confirm the hypervolume unary indicators from Table 1.

The multi-restart PLS, *mPLS*, is the worst performing algorithm. All the other seven (adaptive) genetic PLS are significantly better than it. Indeed, in Figure 2 a), *mPLS* is the worst performing algorithm with the highest values in EAFs. Note the large difference in hypervolume indicators between *mPLS* and the second worst performing algorithm, when compared with the difference between the performance of *rGPLS* and the performance of the six adaptive GPLS algorithms. The best EAFs of the worst performing algorithm *mPLS* have similar performance with the worst EAFs of the

best performing algorithm *aGPLS*. There is a large variance in the attainment surfaces with relatively large standard deviations meaning there are several runs with very good and bad performance. This indicates that bQAP is a very noisy and difficult landscape where standard adaptive algorithms, *aGPLS* and *aBGPLS*, cannot improve on uniformly randomly selecting operators with *rGPLS*.

The bi-operator adaptive pursuit in GPLS gives the algorithm with the best performance from all the tested algorithms. In Figure 2 c), the 2% EAF dominates the best known front *NDA**. Figure 2 d) shows that *gGPLS*'s hypervolume indicator increases faster than for *aGPLS*. *gGPLS* is slightly better than the dual-layer adaptive pursuit GPLS, *tGPLS*, and bi-operator multi-armed bandit GPLS, *gBGPLS*. Both *gGPLS* and *tGPLS* are significantly better than *rGPLS*, the standard adaptive pursuit GPLS *aGPLS*, and two of the multi-armed bandit GPLSs, *aBGPLS* and *tBGPLS*. *gBGPLS* has an in-between performance, being significantly different only to *mPLS*. Looking at the performance of a single run, not presented here because of lack of space, the increase in performance is made in few large jumps.

To understand the difference in behavior between bi-operator and standard adaptive pursuit, Figure 3 shows the percentage of operators used in a randomly selected run by *aGPLS* and *gGPLS*. For both methods, there are several operators, mutation and path guided mutation, that have a larger probability of selection than uniformly randomly selection operators with probability 0.015. We call these operators active

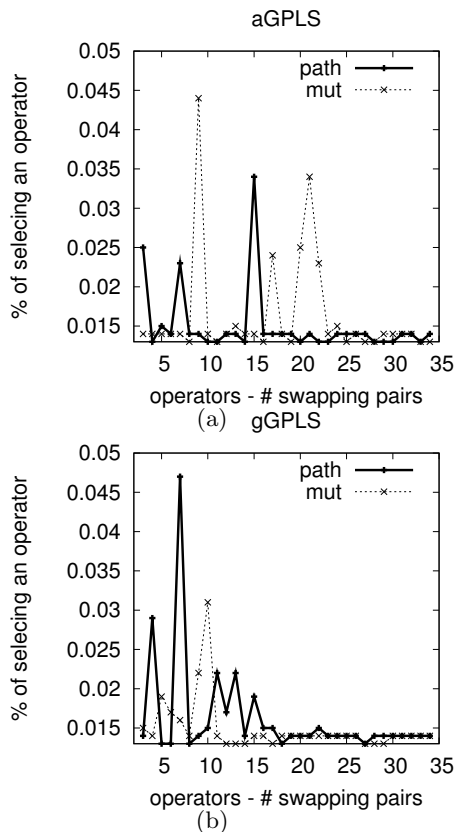


Figure 3: The percentage of using each operator in a random run of a) *aGPLS* and b) *gGPLS*.

operators. The largest probability of an active operator is ≈ 0.045 , about 3 times more than 0.015. *gGPLS* uses path-guided mutation more often than mutation, and *aGPLS*, vice-versa. The active operators for *gGPLS* are focused on the first 1/3 of the x-axis, whereas for *aGPLS* there are active operators with a very large number of exchanges.

Note that the active operators of *gGPLS* perform well for the same bQAP instance in [2]. This indicates that *aGPLS* is stuck a long time with less fitted operators whereas *gGPLS* quicker finds good operators. As a general observation, bQAP with high correlation is a suitable test problem for adaptive strategies, because: a) similar operators generate GPLS with similar performance, b) there are groups of operators that generate GPLSs with very high or very low performance.

6. CONCLUSION

We have proposed a generalization of the adaptive pursuit algorithm that allows any target distribution for operator selection. We call it the generalized adaptive pursuit algorithm (GAPA). We have shown that the probability matching and multi-armed bandit adaptive selection operator techniques can also be considered within the GAPA framework. We propose new techniques to generate target distributions for GAPA designed for a large number of related operators to select from. The multi-operator AP pursues more than one operator assuming that this set of pursuit operators is more advantageous than pursuing only one of these operators. For k-operator GAPA, the number of pursuit operators is fixed and known a priori. The power distribution GAPA uses the

power 2 distribution as the target distribution. Multi-layer GAPA allows to discriminate between parameters with good and poor performance. It combines different GAPA units with different target distributions and learning parameters. Dual-layer GAPA is scaling up the adaptation because the three GAPAs contain less than half of the number of operators than a single layer GAPA does. Dynamical multi-layer GAPA learns blocks of well performing operators in GAPAs of different sizes and settings. Because the probability matching and multi-armed bandit can be integrated in GAPA, the proposed instances of GAPA are also applicable to them. We therefore claim that GAPA is a framework in which new adaptive selection strategies can be generated.

We use GAPA to build adaptive GPLS (aGPLS) algorithms for multi-objective QAPs. We design update rules to adaptively select between mutation and path-guided mutation genetic operators with different swapping rates. The experimental results on bi-objective QAPs with a large number of facilities and high correlations between the dimensions reveal the standard multi-restart PLS as the worst performing algorithm. The genetic PLS which uniform randomly samples from a range of exchange rates for mutation and path-guided mutation can be outperformed by adaptive GPLS. The bi-operator *gGPLS* and dual-layer *tGPLS* are the best performing algorithms by pursuing two operators instead of one operator like the standard adaptive pursuit algorithm. We conclude that the multi-operator and multi-layer adaptive operator selection techniques can be efficiently used in designing adaptive strategies.

7. REFERENCES

- [1] L. DaCosta, A. Fialho, M. Schoenauer, and M. Sebag. Adaptive operator selection with dynamic multi-armed bandits. In *Proc. of Conf. of Genetic and Evol. Comp., GECCO'08*, pages 913–920. ACM, 2008.
- [2] M. Drugan and D. Thierens. Path-guided mutation for stochastic Pareto local search algorithms. In *Proc. of Parallel solving problems from Nature, PPSN'10*, pages 485–495. LNCS 6238, Springer, 2010.
- [3] J. Knowles and D. Corne. Instance generators and test suites for the multi-objective quadratic assignment problem. In *Proc. of Evol. Multi-Criterion Optim. (EMO)*, number 2632 in LNCS, pages 295–310, 2003.
- [4] L. Paquete and T. Stutzle. A study of stochastic local search algorithms for the biobjective QAP with correlated flow matrices. *European J. of Oper. Res.*, 169(3):943–959, 2006.
- [5] D. Thierens. An adaptive pursuit strategy for allocating operator probabilities. In *Proc. of Conf. on Genetic and Evol. Comp., GECCO'05*, pages 1539–1546, 2005.
- [6] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V.G. da Fonseca. Performance Assessment of Multi-objective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.