A Comparison of Adaptive Operator Scheduling Methods on the Traveling Salesman Problem

Wouter Boomsma

Department of Computer Science,
University of Aarhus, Denmark
wb@daimi.au.dk

Abstract. The implementation of an evolutionary algorithm necessarily involves the selection of an appropriate set of genetic operators. For many real-world problem domains, an increasing number of such operators is available. The usefulness of these operators varies for different problem instances and can change during the course of the evolutionary process. This motivates the use of adaptive operator scheduling (AOS) to automate the selection of efficient operators. However, little research has been done on the question of which scheduling method to use. This paper compares different operator scheduling methods on the Traveling Salesman Problem. Several new AOS techniques are introduced and comparisons are made to two non-adaptive alternatives.
The results show that most of the introduced algorithms perform as well as Davis' algorithm while being significantly less cumbersome to implement. Overall, the use of AOS is shown to give significant performance improvements – both in quality of result and convergence time.

1 Introduction

Genetic operators are the algorithmic core of evolutionary computation. The quality of these operators is essential for the performance of the evolutionary algorithms (EAs), and consequently, much research is done in this field. Particularly for combinatorial problems, domain knowledge is often essential to good performance. Many heuristic operators have been described, and their number continues to grow. Given their heuristic nature, the applicability of these operators often varies across different problems in the domain. For instance, certain operators might fail to scale up gracefully, becoming computationally infeasible for larger problem instances. Even though the literature might provide comparisons between operators on certain problem instances, the evolutionary programmer is left with a difficult choice of operator selection when designing an EA for a new problem.

An elaborate manual comparison of all operators would provide an assessment of the quality of the operators, but does not exploit the fact that the usefulness of the operators often changes during the course of a single run. Furthermore, dependencies between operators can exist and through interaction multiple operators might provide better results than when applied alone.

In a previous study [1], I investigated whether adaptive operator scheduling (AOS) can provide a solution to this problem. Experiments were done on instances of the symmetrical Traveling Salesman Problem (TSP), a well known NP-hard combinatorial problem for which a multitude of operators exist. It can be defined as the search for a minimal Hamiltonian cycle in a complete graph, and can be understood as the problem of visiting n cities (and returning to the first), using the path of smallest total distance. The main concern in the original investigation was that a large number of operators might slow down the optimisation process compared to an algorithm using the optimal choice of operators. It was shown that this concern was unfounded: the algorithm using AOS converged as fast as the best combination of mutation and crossover operators with equally good results. The AOS scheme used in these initial experiments (Davis) was however rather cumbersome to implement. In this paper several alternative methods of AOS are presented and compared on a selection of TSP benchmark problems. It is shown that the presence of multiple operators significantly improves performance. Furthermore, results indicate that the alternative AOS methods perform as well as Davis' method, and that simpler methods can thus be used without a decrease in performance.

2 Adaptive Operator Scheduling

Angeline [2] has categorised adaptive evolutionary computation based on two criteria: (1) the level at which adaptation is applied and (2) the nature of the update rules. The level of adaptation specifies at which level the adapted parameters reside. For *population-level* adaptation, the parameters exist at the population level, and thus apply to all individuals in the population. Likewise, *individual-level* adaptation works on parameters local to each individual and *component-level* adaptation has parameters for each component in the genotype of an individual. Update rules are classified as being *absolute*, or *empirical*. This last class also goes by the name of *self-adaptation*, which is the term that will be used throughout this paper. Algorithms with absolute update-rules use fixed guidelines to update parameter settings based on the current state of the population. Self-adaptive schemes use the selection pressure already present in the evolutionary process to evolve better parameter settings, the underlying assumption being that good individuals often have a good parameter setting.

The AOS algorithms described below represent different classes of Angeline's classification. Focus was on designing AOS algorithms that are easily implemented, which is vital if AOS is to replace the manual comparison of individual operators. The end of each description contains a list of the parameter settings that were used for the algorithm. These settings were manually tuned based on preliminary experiments.

2.1 The Operator Scheduling Algorithm by Davis

Davis' algorithm from 1989 [3] represents one of the first efforts at operator adaptation in EAs. It uses population-level adaptation and absolute update rules

on a steady state EA. More specifically, a global set of operator probabilities is adapted based on the performance of the operators in the last generations (adaptation window). The performance of operators is measured by the quality of the individuals they produce. Newly created individuals are rewarded if their fitness surpasses the fitness of all other individuals in the population. The size of this reward is determined by the amount of the improvement. Furthermore, a certain percentage of the reward is recursively passed on to the individual's ancestors (to a certain maximum depth M). This reward strategy is motivated by the fact that a series of suboptimal solutions is often necessary in order to reach a better solution, and that corresponding operators should thus be rewarded.

With certain intervals, the rewards are used to update the probability setting. For each operator i:

$$p'_i = (1 - S) * p_i + S * \frac{\text{reward}_i}{\text{totalReward}}, \tag{1}$$

where p'_i is the new probability for operator i and S is the Shift factor, determining the influence that the current update should have on the total probability setting.

In the present study a slightly modified version of Davis' algorithm is used. Lower bounds are set on the probabilities to avoid extinction of operators, and in adaption phases where no improvement is made the probabilities are shifted slightly toward their initial positions. Furthermore, the steady state evolutionary approach is replaced by a generational elitist EA. Preliminary experiments showed that these modifications gave better results and faster convergence (results not shown).

Parameter settings: Window of adaptation (W): 100 individuals, Interval of adaptation (I): 20–50 evaluations, Shift percentage (S): 15%, Percentage of reward to pass back (P): 90%, Number of generations to pass back (M): 10.

2.2 The ADOPP Algorithm

Julstrom's Adaptive Operator Probabilities algorithm (ADOPP) [4] from 1995 is very similar to Davis' approach. The main difference between the two is the way in which ancestral information is represented. While Davis provides each individual with links to their parents, Julstrom explicitly provides each individual with a tree specifying which operators were used to create its ancestors. Davis' approach is more effective since the tree structure is implicitly present in the individuals and does not have to be copied every time new individuals are created. However, Davis' algorithm has some disadvantages which Julstrom avoids. Since the ancestral information in Davis' algorithm is stored in the individuals, information is lost when individuals die. The depth of the implicit trees are therefore somewhat unreliable and, in general, Davis' algorithm is only able to maintain a moderate amount of ancestral information. Furthermore, the individuals in Davis' algorithm require a great deal of bookkeeping to sustain pointers only to living individuals, an inconvenience which is avoided in ADOPP.

Another difference between the two approaches is that Davis rewards individuals that improve the overall best individual in the population, whereas Julstrom only requires individuals to exceed the median individual.

When converting operator rewards to a new probability setting, ADOPP uses a greedy variant of the update rule by Davis (corresponding to S=100%). For each operator i:

$$p'_i = \frac{\text{reward}_i}{\text{totalReward}}, \tag{2}$$

where p'_i is the new probability for operator i.

Parameter settings: Window of adaptation ($QLEN$): 100 individuals, Interval of adaptation (I): 1 generation, Percentage of reward to pass back ($DECAY$): 80%, Height of trees ($DEPTH$): 4.

2.3 Adaptation Using Subpopulations (Subpop)

This approach was inspired by the work by Schlierkamp-Voosen and Mühlenbein [5] on competing subpopulations, in which different subpopulations represent different operator strategies. This idea can be used as an AOS method by letting each subpopulation represent the use of a single operator. The EA thus maintains a number of subpopulations equal to the number of operators. For each of these populations, only one operator can be applied to the individuals currently residing there. During the course of a run, the relative sizes of the subpopulations are altered depending on their fitness. The fitness of a subpopulation is based on the fitness of its best individuals over the last 10 generations (as proposed by Schlierkamp-Voosen and Mühlenbein [5]).

In each adaptation phase the best subpopulation is rewarded with an increase in size, and receives a donation of individuals from all other subpopulations. To avoid the extinction of operators, only subpopulations with a size above some fixed lower bound are forced to make this donation. Large subpopulations create more offspring and thus have a larger probability of improving the global best fitness. This results in a strong bias towards larger subpopulations, making it difficult for smaller populations to compete. To counter this effect, a random migration scheme is used: At certain intervals some of the best individuals from the best population migrate to a randomly selected population.

Subpop is a population-level adaptation algorithm using absolute update rules. The parameters adapted are the same as for Davis and ADOPP (global operator probabilities).

Parameter settings: Evaluation interval (E): 4 generations, Shift amount (SA): 10%, Migration interval (M): 4 generations, Migration amount (MA): 5 individuals, Interval of adaptation (I): 1 generation.

2.4 Self-Adaptive Operator Scheduling (SIDEA, SPDEA)

The algorithms in this section are novel methods of operator adaptation partly inspired by the work of Spears in 1995 [6]. In Spears' paper, scheduling is done

between two crossover operators by adding a single bit to the genotype indicating the preferred operator. This bit is used either locally (at the individual-level) or globally (at the population-level) to determine which operator is to be applied. When used locally, the choice of operator is made based on the bit-setting of the parent(s) involved. When used globally, the average bit-setting of the whole population is used as a measure of quality to base this decision on.

I generalised Spears' method by expanding the single bit to an array of n probabilities, each denoting the probability that a certain operator is applied. Unlike Spear's approach, the representation of the scheduling information is not compatible with that of the problem representation and therefore cannot be modified automatically as part of the genotype. It was therefore necessary to design a specialised variation operator for this task.

The problem of finding the optimal probability setting for n operators is a numerical optimisation problem so in principle, any variation operator from this domain would suffice. However, given the fact that the probabilities must all be positive and sum to one, only a small subset of the n-dimensional search space constitutes legal solutions. If an arbitrary EA variation operator is used it would be necessary to apply some repair scheme after each operation to ensure feasibility. To avoid this I used an adapted version of the variation operator used in Differential Evolution (DE) [7], which produces solutions that are only rarely illegal. The DE variation operator uses three individuals to create an offspring by applying a mutation step followed by a crossover step. During the mutation step, a new genotype $\boldsymbol{x}'$ is created by:

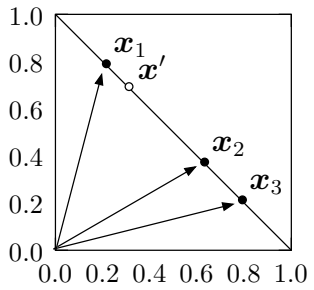

Fig. 1. The DE mutation operator for two dimensions. Three random individuals $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{x}_3$ are chosen and a temporary individual $\boldsymbol{x}'$ is created by taking the vectorial difference between $\boldsymbol{x}_3$ and $\boldsymbol{x}_2$, multiplying it by F, and adding it to $\boldsymbol{x}_1$.

$$\boldsymbol{x}' = \boldsymbol{x}_1 + F * (\boldsymbol{x}_3 - \boldsymbol{x}_2) \ . \tag{3}$$

The crossover step subsequently creates a new individual by combining components from $\boldsymbol{x}'$ with components from the original individual. For our purpose, however, the mutation step alone has exactly the properties we need: Selecting three random points on a hyperplane and adding the vectorial difference between two of them to the third results in a new position on the plane. In other words, given three legal probability settings it will produce a new one with probabilities that sum to 100%. An example of this behaviour for two dimensions is given in Fig. 1. Equation (3) of course gives no guarantees that the entries of the created vector have values between zero and one. This is however easily fixed by forcing values that lie outside the domain back to the nearest boundary. The mutation operator is used whenever new offspring is to be created.

As with Spears' algorithm, two different levels of adaptation exist. If individual-level adaptation is used, the probability setting of the child is used to choose an operator. If adaptation works at the population-level, the average of all probability-settings in the population is used to select an operator. The two versions of the algorithm will be referred to as the SIDEA and the SPDEA respectively.

Parameter setting: F-value for the DE mutation: 0.5.

2.5 Operator Scheduling Inspired by Evolution Strategies (SIESA)

Self-adaptation has been applied in the Evolution Strategies (ES) community since 1977 [8]. Here it was used to adapt the behaviour of the mutation operator during the course of a run. The mutation operator consists of an addition of normally distributed values $z_i \sim \mathbf{N}(0, \sigma_i'^2)$ to the components in the genotype. The variances $\sigma_i'^2$ are adapted so that the effect of mutation is different for different individuals and for different components of the genotype. Again, under the assumption that individuals of high fitness often have good parameter settings, the algorithm can find the good variance setting using only the selection pressure already present in the algorithm.

Taking self-adaptation one step further, we now use the ES mutation step as an alternative to the DE mutation operator described in the previous section. This means that the evolutionary algorithm will simultaneously optimise the problem at hand, an operator probability setting, and a setting determining the optimal variance for the mutation of this probability setting.

Since the search space defined by the operator probability settings is highly interdependent, it is meaningless to adapt each variance σ_i^2 at component-level. Instead one value σ is associated with each individual. The necessary mutation equation from the ES literature [9] is

$$\sigma' = \sigma * \exp(s_0), \tag{4}$$

where $s_0 \sim \mathbf{N}\left(0, \tau_0^2\right)$ and $\tau_0^2 = \frac{1}{n}$. Since this mutation operator does not have the convenient properties of the DE operator, the probability-setting has to be normalised after each mutation. It is difficult to tell exactly how big the impact of this normalisation is compared to the effect of the mutation itself.

Parameter setting: τ_0^2: $\frac{1}{n}$ (ES default value [9]).

3 Operators

Table 1 lists the 18 operators used in the experiments. They all operate on a path representation of the TSP. The selection of operators was inspired by Larrañaga's survey paper from 1999 [10]. The list was extended by two operators of recent date that have been shown to have good performance: The Edge Assembly Crossover [11] and the Inver-over operator [12]. For complete references the reader is referred to the paper covering my previous experimentation on AOS for the TSP [1].

Table 1. The Operators

Displacement Mutation (DM)	Order Based Crossover (OX2)
Exchange Mutation (EM)	Position Based Crossover (POS)
Insertion Mutation (ISM)	Heuristic Crossover (HEU)
Simple Inversion Mutation (SIM)	Edge Recombination Crossover (ER)
Inversion Mutation (IVM)	Maximal Preservative Crossover (MPX)
Scramble Mutation (SM)	Voting Recombination Crossover (VR)
Partially mapped Crossover (PMX)	Alternating Position Crossover (AP)
Cycle Crossover (CX)	Inver-over operator (IO)
Order Crossover (OX1)	Edge Assembly Crossover (EAX)

4 Experiments and Results

Initial experiments showed that the algorithms benefitted from larger populations as the problem instances grew, confirming the findings by Nagata and Kobayashi [11]. The large populations are however only partly replaced in each generation. The ADOPP variant uses a steady state approach, generating only one new individual in each generation (in agreement with Julstrom's original paper [4]). The other algorithms were run with elite sizes of 75% – 90% of the population, which proved to give best results in initial experiments. With these extensive elite sizes the large populations function mainly as libraries of genetic material which maintain a certain diversity in the population. The population sizes and number of iterations used are listed in Table 2.

Table 2. Population sizes and iterations used for the different problems

	gr48	brg180	pcb442	nrw1379	pr2392	pcb3038
Population Size	500	600	2000	4000	4500	5000
Iterations	60,000	350,000	500,000	1,200,000	1,200,000	1,700,000

When implementing an AOS algorithm, one has the choice of including all operators in one pool or dividing mutation and crossover operators in two separate pools. For Davis, ADOPP, SIDEA and SPDEA, both variants were implemented and included in the test set. The versions using separate pools are labelled with the suffix _S. Two non-adaptive algorithms were included in order to evaluate the absolute value of AOS. The Uniform algorithm includes all operators but uses equal probabilities for the application of them. The EAX_only uses only the EAX operator.

For all algorithms, 100 runs were done on 6 different TSP instances, all taken from the TSP benchmark problem collection TSPLIB [13]. The sizes of the problems ranged from 48 to 3038 cities. Table 3 presents the average results of these trials.

To give an impression of convergence speed Fig. 2 shows the average best fitness over time for the brg180 problem. The general pattern of this figure is recurrent across the range of tested problems: The EAX_only algorithm performs

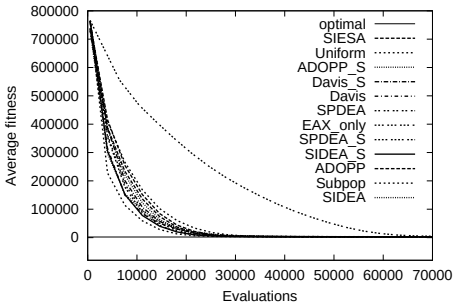

Fig. 2. Convergence graph for all algorithms on the brg180 problem

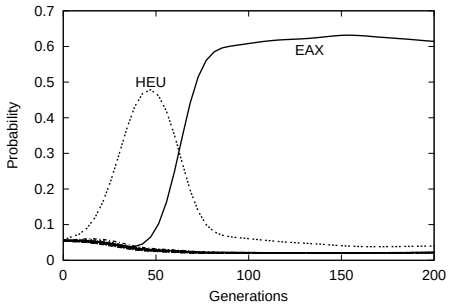

Fig. 3. Operator probability distribution during an execution of SIDEA on brg180

significantly worse than all others, the Uniform algorithm performs almost as good as the adaptive algorithms, and among the adaptive algorithms, Subpop tends to have the fastest convergence. Figure 3 shows the operator probabilities over time for the SIDEA algorithm applied on the brg180 problem. This figure is also representative for all cases: Across both problem instances and AOS methods all algorithms consistently prioritise the EAX operator in the final phase. In the initial phases the greedy nature of the other operators (typically HEU) is most effective to gain fast fitness improvements. This clearly improves performance compared to the algorithm applying the EAX operator exclusively.

Based on the data in Table 3, it is not possible to single out one algorithm as being the overall best. However, a certain categorisation of the algorithms can be made. Across the range of problems, the Davis(_S), SIDEA(_S), SPDEA(_S) and Subpop algorithms all performed about equally well. The data responsible for the mean values in the table were found to be approximately normally distributed. Standard deviations and confidence intervals for the results were computed. Although there were non-overlapping confidence intervals between results in Table 3, and differences thus can be seen as being statistically significant, it varies from problem to problem which algorithm performed best.

The ADOPP and SIESA algorithms performed slightly worse than this first class. For ADOPP's case this might be ascribed to the algorithm being more greedy than, for instance, the Davis algorithm, and therefore might overcompensate bad operators when they by chance improve the overall solution. On the other hand, it seems that the SIESA adapts too slowly. This is not entirely surprising, since it is using self-adaptation at two levels. Another reason could be the somewhat disruptive renormalisation after each operation.

Perhaps the most striking results in Table 3 concern the two non-adaptive algorithms. The EAX_only algorithm performs significantly worse than any of the other algorithms. This suggests that the EAX operator is first and foremost a fine-tuning operator and should be used along with some more exploration-oriented operators. The Uniform algorithm, on the other hand, performs remarkably well. Especially the convergence graphs show surprisingly fast convergence.

It should however be noted that the convergence graphs show the population's best fitness over time and that no information about the general state of the population can be derived from these graphs. The apparent conflict between the Uniform algorithm's good convergence speed and the somewhat worse end-results of this algorithm indicates that even though the algorithm is able to sustain a good elite of individuals, it does not have sufficient diversity in its best individuals to find optimal solutions. The smaller number of good individuals is naturally explained by the fact that in the end phase, the EAX operator is applied with only a fraction of the probability that it receives by the adaptive algorithms.

Generally, the adaptive operator scheduling methods did not present a significant computational burden for the EAs involved. The different algorithms had similar execution times, except for the ADOPP variants, which were somewhat slower due to their steady state design.

Table 3. Average results for 100 runs

	gr48	brg180	pcb442	nrw1379	pr2392	pcb3038
Davis_S	5046.00	1950.30	50778.21	56689.46	378067.16	137758.94
Davis	5046.36	1950.00	50781.23	56685.04	378055.16	137752.69
SIDEA_S	5047.77	1950.20	50778.14	56699.82	378084.82	137779.77
SIDEA	5046.52	1950.30	50778.07	56715.85	378336.45	137793.97
SPDEA_S	5046.33	1950.40	50778.21	56698.31	378097.53	137781.66
SPDEA	5046.00	1950.30	50778.07	56691.05	378074.36	137755.16
SIESA	5047.21	1952.60	50781.83	56695.51	378098.00	137765.02
Subpop	5046.40	1950.60	50778.14	56682.53	378057.28	137743.03
Uniform	5048.34	1952.80	50778.91	56826.32	379570.91	137956.90
ADOPP_S	5046.66	1951.80	50778.56	56702.35	378100.71	137779.00
ADOPP	5046.66	1951.70	50778.35	56689.42	378079.92	137745.00
EAX_only	5046.00	1951.30	59558.96	128619.73	2823286.85	677724.28
Optimal	5046.00	1950.00	50778.00	56638.00	378032.00	137694.00

5 Discussion

Overall, the results show that performance is significantly improved when a multitude of operators are used. Even when one operator (e.g. EAX) is suspected to outperform all others, it might not be optimal throughout the whole run, and adaptive operator scheduling can exploit this fact to increase performance.

Based on the comparison of operator scheduling algorithms in this study, it is not possible to single out one of them as being the best. Many of the different methods performed equally well, and thus indicate that the selection of an operator scheduling method may be based on considerations as implementation efficiency or personal taste. The Davis algorithm requires significant bookkeeping and was found to be somewhat elaborate to implement, while especially the SIDEA, SPDEA, SIESA and Subpop variants were implemented fairly easily.

Also the fact that the self-adaptive variants have only few parameters to tune might be a reason to favour these over the others. As an extreme case, even a uniform selection between operators seems to provide reasonable results at minimal implementation costs. Naturally, experiments should be carried out on other problem domains to determine the value of operator scheduling in general and establish whether the above conclusions hold for other domains.

References

1. Boomsma, W.: Using adaptive operator scheduling on problem domains with an operator manifold: Applications to the travelling salesman problem. In: Proceedings of the 2003 Congress on Evolutionary Computation (CEC2003). Volume 2. (2003) 1274–1279
2. Angeline, P.J.: Adaptive and self-adaptive evolutionary computations. In Palaniswami, M., Attikiouzel, Y., eds.: Computational Intelligence: A Dynamic Systems Perspective. IEEE Press (1995) 152–163
3. Davis, L.: Adapting operator probabilities in genetic algorithms. In Schaffer, J.D., ed.: Proceedings of the Third International Conference on Genetic Algorithms, San Mateo, CA, Morgan Kaufman (1989)
4. Julstrom, B.A.: What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm. In Eshelman, L., ed.: Proceedings of the Sixth International Conference on Genetic Algorithms, San Francisco, CA, Morgan Kaufmann (1995) 81–87
5. Schlierkamp-Voosen, D., Mühlenbein, H.: Strategy adaptation by competing subpopulations. In Davidor, Y., Schwefel, H.P., Männer, R., eds.: Parallel Problem Solving from Nature – PPSN III, Berlin, Springer (1994) 199–208
6. Spears, W.M.: Adapting crossover in evolutionary algorithms. In McDonnell, J.R., Reynolds, R.G., Fogel, D.B., eds.: Proc. of the Fourth Annual Conference on Evolutionary Programming, Cambridge, MA, MIT Press (1995) 367–384
7. Storn, R., Price, K.: Differential evolution - a simple and efficient heuristic for global optimisation over continuous spaces. Journal of Global Optimization **11** (1997) 341–359
8. Bäck, T., Hoffmeister, F., Schwefel, H.P.: A survey of evolution strategies. In Belew, R., Booker, L., eds.: Proceedings of the Fourth International Conference on Genetic Algorithms, San Mateo, CA, Morgan Kaufman (1991) 2–9
9. Bäck, T.: Evolution strategies: An alternative evolutionary algorithm. In Alliot, J., Lutton, E., Ronald, E., Schoenhauer, M., Snyers, D., eds.: Artificial Evolution, Springer (1996) 3–20
10. Larrañaga, P., Kuijpers, C.M.H., Murga, R.H., Inza, I., Dizdarevic, S.: Genetic algorithms for the travelling salesman problem: A review of representations and operators. Artificial Intelligence Review **13** (1999) 129–170
11. Nagata, Y., Kobayashi, S.: Edge assembly crossover: A high-power genetic algorithm for the travelling salesman problem. In Bäck, T., ed.: Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97), San Francisco, CA, Morgan Kaufmann (1997)
12. Tao, G., Michalewicz, Z.: Inver-over operator for the TSP. In Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.P., eds.: Parallel Problem Solving from Nature – PPSN V, Berlin, Springer (1998) 803–812
13. Reinelt, G.: TSPLIB — a traveling salesman problem library. ORSA Journal on Computing **3** (1991) 376–384