Using adaptive operator scheduling on problem domains with an operator manifold: Applications to the Travelling Salesman Problem

Wouter Boomsma
EVALife group, Dept. of Computer Science, University of Aarhus
Ny Munkegade, Bldg. 540, DK-8000 Aarhus C, Denmark
wb@daimi.au.dk

Abstract- A growing problem in the field of evolutionary computation is the large amount of genetic operators available for certain problem domains. This tendency is especially pronounced in areas where heuristics are used to create highly specialised operators. Even within the same problem domain, the performance of such operators often depends on the specific problem instance at hand. This results in a tedious and time-consuming process of comparing individual operator performances every time a new problem is to be solved.

This paper investigates the use of adaptive operator scheduling to automate the operator selection process. The approach is tested on instances of the Travelling Salesman Problem - a problem for which a long list of operators exists. Results show that benefits are twofold: Operator selection is achieved automatically and an overall performance improvement is observed.

1 Introduction

It has become generally accepted that genetic algorithms benefit from the incorporation of domain specific knowledge. Many real-world problems are solved by designing a problem specific representation and corresponding operators that manipulate individuals with methods inspired by the context of the problem. The possibilities for creating more or less sophisticated heuristic operators are endless. Papers presenting new operators for specific problem domains appear frequently.

When confronted with a new problem to solve, the evolutionary programmer is thus forced to take a pick in a large pool of operators. Even though the literature will contain comparisons between certain selections of operators, the uncertainty remains: Which combination of operators performs best for the specific problem at hand? A good overview requires the programmer to invest time and effort in comparing all operators individually.

In this paper I investigate whether adaptive operator scheduling is a possible solution to this problem. The idea is that the whole set of candidate operators is implemented, and that the optimal choice of operators is found automatically by the algorithm.

Previous work is not conclusive on the effects that adaptive operator scheduling has on performance. Davis presents good results on neural networks in his original paper from 1989[1]. Srinivas & Patnaik tested an approach on both numerical benchmark problems and smaller instances of the TSP with good results on most problems[4]. Julstrom does well on the same problems with a different approach[5]. Thomsen & Krink achieve encouraging results on a set of numerical benchmark functions[2]. However, Eiben, Sprinkhuizen-Kuyper & Thijssen report that performance was not improved in their competing crossover approach[3], and Tuson & Ross only achieved better performance on certain problems[6]. In the light of these results, significant performance improvements over non-adaptive algorithms are not anticipated. The adaptive approach might introduce some improvements as an added bonus, but solving the tedious operator selection problem is the main objective.

The algorithm is tested on the travelling salesman problem - a well known NP-hard combinatorial problem for which many operators have been designed. TSP consists of determining the shortest Hamiltonian cycle in a complete graph, i.e. visiting all nodes and minimising the summed weight of the edges passed on the way.

The paper is organised as follows: Section 2 presents a selection of operators that have been developed for the TSP during the last 30 years. In section 3 the adaptive operator scheduling genetic algorithm (AOSGA) is described followed by a description of the conducted experiments in section 4. Section 5 focuses on a performance comparison between the AOSGA and the genetic algorithms with a fixed combination of operators (fixed-op GAs). I show that that the adaptive approach gives robust results equal or slightly better than any combinations of fixed operators. A discussion of the results and some concluding remarks are presented in section 6.

2 The Operators

Table 1 lists the 6 mutation operators and 10 crossover 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[21]. In recent years other operators have been proposed (e.g. [22, 23, 24]), and some of these are known to outperform the ones listed in table 1. Time has not yet permitted me to implement these more advanced operators. This is however not a serious limitation.

Name	Authors
Partially mapped Crossover (PMX)	Goldberg and Lingle [7]
Cycle Crossover (CX)	Oliver, Smith and Holland [8]
Order Crossover (OX1)	Davis [9]
Order Based Crossover (OX2)	Syswerda [10]
Position Based Crossover (POS)	Syswerda [10]
Heuristic Crossover (HEU)	Grefenstette [11]
Edge Recombination Crossover (ER)	Whitley, Timothy and Fuquay [12]
Maximal Preservative Crossover (MPX)	Mühlenbein, Gorges-Schleuter and Krämer [13]
Voting Recombination Crossover (VR)	Mühlenbein [14]
Alternating Position Crossover (AP)	Larrañaga, Kuijpers, Poza, and Murga [15]
Displacement Mutation (DM)	Michalewicz [16]
Exchange Mutation (EM)	Banzhaf [17]
Insertion Mutation (ISM)	Fogel [18]
Simple Inversion Mutation (SIM)	Holland [19]
Inversion Mutation (IVM)	Fogel [20]
Scramble Mutation (SM)	Syswerda [10]

Table 1: The operators

Since the goal of this paper is to analyse costs and benefits of an adaptive approach compared to an approach with fixed operators, we are interested in relative performance between the approaches rather than the absolute performance of single operators.

3 The AOSGA model

The AOSGA (figure 1) is based on an algorithm presented by Lawrence Davis in 1989[1]. His paper represents one of the first efforts to incorporate adaptive operator scheduling in a genetic algorithm. The idea is as follows: All individuals in the population keep track of the operator used to create them and which other individuals (parents) were involved. New individuals are rewarded if their fitness exceeds the current best fitness in the population. Furthermore, the individual's ancestors are recursively rewarded (to depth `M`) with some percentage `P` of their child's reward. Once every `I`'th generation, every operator is rewarded by the sum of the rewards of the individuals that it has produced. Based on these operator rewards, the new probability settings are computed. The parameter `W` determines the window of adaptation (how many of the recently created individuals should be considered) and the parameter `S` determines how much the probability is shifted. The probability update equation for the ith operator in the operator pool is:

$$\mathrm{p}'[i] = (1 - \mathrm{S}) * \mathrm{p}[i] + \mathrm{S} * \frac{\mathrm{reward}[i]}{\mathrm{totalReward}} \qquad (1)$$

A few modifications were made to Davis' original description. First, the steady state approach was generalised to a generational GA with a variable elite size (Setting the elite size to one below the population size gives us Davis' steady state approach). A generation consists of recombination, mutation and selection, and for each of these a group of operators is defined. It is within these groups that Davis' probability adaptation scheme is applied (the probabilities in each group thus sum to 1.0).

AOSGA
initialize($population$)
while (!done) {
recombine($population$)
mutate($population$)
select($population$)
if($generations$ mod $I == 0$)
adapt()
}

adapt()
PassOnRewardsToParents()
for each of the W individuals created most recently **do**
assignOperatorRewards()
updateOperatorProbabilities()

Figure 1: Pseudo-code for the AOSGA

A lower bound (2%) on the operator probabilities was introduced to avoid that operators go extinct during the run. This is done to ensure that all operators have a chance of revival after a period of dominance of a rival operator. To further encourage this recovery, the operator probability setting slowly expires: If the AOSGA completes a generation

TSP	Optimum	Evaluations used	Best fixed-op GA			AOSGA	
			Operators	Best	Average	Best	Average
gr48	5,046	50,000	IVM+HEU	**5,046**	5,184.8(2.6)	**5,046**	5,166.0(2.4)
brg180	1,950	2 mil.	IVM+OX1	1,980	2,036.2(3.2)	1,970	2,067.0(4.6)
pcb442	50,778	3 mil.	SIM+OX2	54,847	57,278(91)	53,532	55,442.0(81)

Table 2: Best fixed-op GA vs. the AOSGA

without improving the best solution, it adjusts the operator probabilities slightly towards the original values. The probability update function is in this case:

$$\mathrm{p}'[i] = (1 - 0.005) * \mathrm{p}[i] + 0.005 * \frac{1}{\mathrm{NrOfOps}} \qquad (2)$$

where NrOfOps is the number of operators in the pool.

4 Experiments

This study was motivated by a wish to avoid the tedious operator selection process when faced with a new problem. Two criteria must be fulfilled for this approach to be successful:

1. Quality: The results of the AOSGA should be as good as the best of the fixed-op versions.

2. Speed: Convergence speed should not be significantly reduced.

The quality issue is investigated by comparing the results of the AOSGA with all fixed-op combinations of a single mutation and crossover operator. Speed is tested by plotting the current best solution as a function of used evaluations during the course of the solution process (averaged over some amount of runs).

Experiments were conducted on 3 symmetrical TSP instances: A 48-city problem, a 180-city problem and a 442-city problem (gr48, brg180 and pcb442), all of which are taken from the TSP benchmark problem collection TSPLIB[25].

Some initial experimentation was done to determine the best parameter settings for the algorithm. A crossover rate of 0.6 and an overall mutation rate of 1.0 were used. This is only reasonable if the elite size is kept relatively high - 80 out of a population size of 100 proved to be a good compromise between a classical generational GA and the steady state model used by Davis. Some of the TSP operators have additional parameters which were fine-tuned by hand based on single-operator runs of the algorithm. Details on these parameter settings can be found in the original descriptions of the operators (see table 1). The settings used were:

- OX2 Maximum percentage of genes to move: 0.95
- POS Maximum percentage of genes to fix: 0.05
- VR Maximum percentage of population that can be chosen as parents: 0.25
  Vote threshold: 0.05
- SM Maximum size of random sub-tour (in percentage of genome size): 0.5

All operator probabilities were initially set to $1/\mathrm{NrOfOps}$. The parameters involved in controlling the adaptation scheme were set as follows:

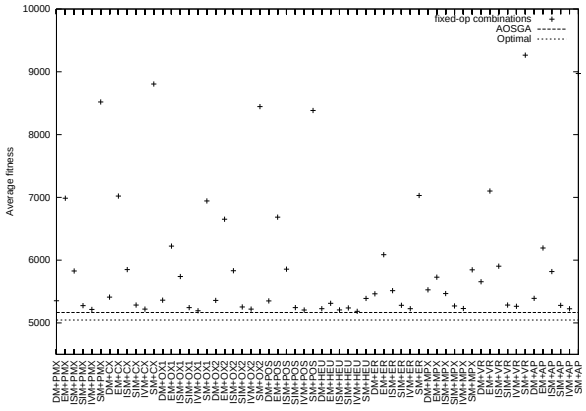
P	S	W	M	I
90%	15%	100	10	1

which were inspired by the values that Davis proposed. The robustness of the algorithm to variations in these setting is however rather high, and other values gave equally good results.

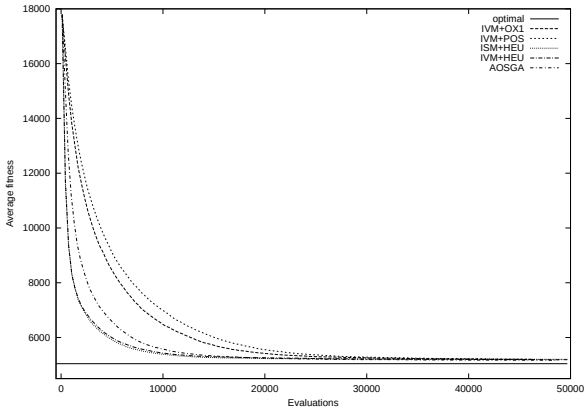
5 Results

All pairwise combinations of mutation and crossover operators were applied to the 3 TSP problems. For each of these combinations, a simple tournament selection scheme was used. In table 2 the combination with the best average performance is compared with the results of the AOSGA using all operators. The results on gr48 are based on 1000 repetitions, while only 100 repetitions were conducted for the two other problems. Mean values are given with the standard error in parentheses. Execution time ranged from 3 seconds for the gr48 problem to 6-7 minutes for the pcb442 problem (Processor: Xeon 1.7 GHz). A graphical overview of the comparisons is presented in figure 2.

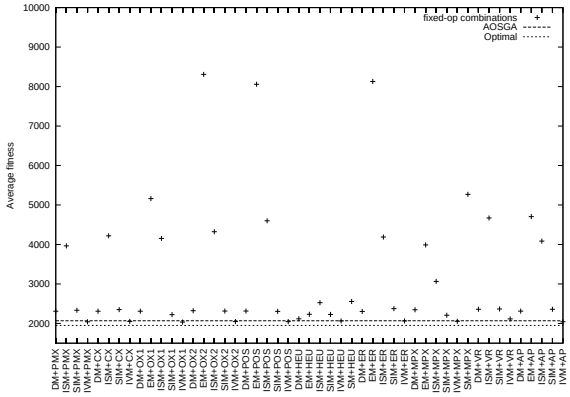
To get an impression of the convergence speed of the AOSGA, progress during the course of a run was logged and compared to the progress of 4 of the best performing fixed-op algorithms. Results are found in figure 2(d-f). The graphs indicate that convergence speed of the AOSGA is not worse than that of the average fixed-op algorithm. For the larger problems, it would even seem that the AOSGA reaches acceptable solutions somewhat faster than the fixed-op algorithms.



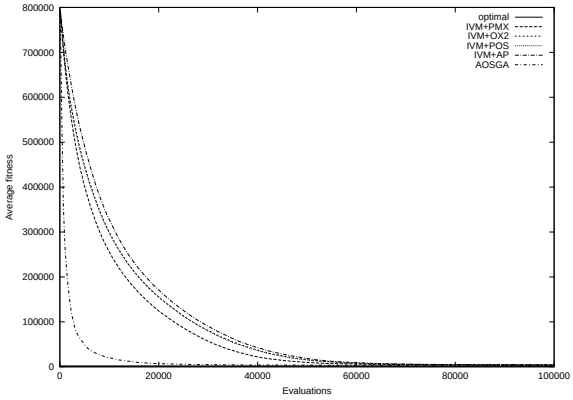
(a) gr48



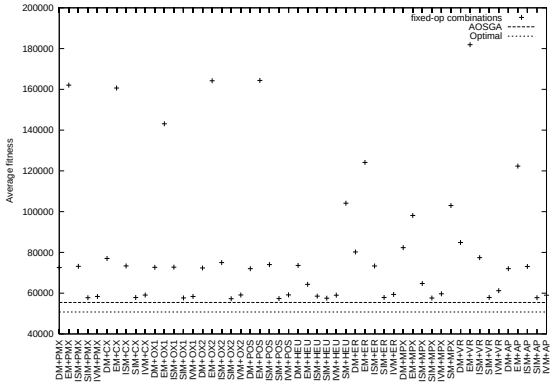
(d) gr48



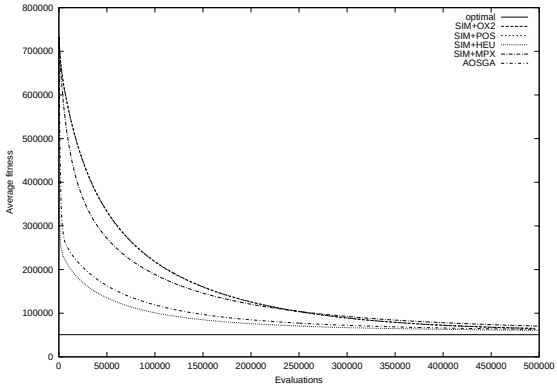
(b) brg180



(e) brg180



(c) pcb442



(f) pcb442

Figure 2: Fixed-op combinations vs. the AOSGA. (a-c) show the best fitness averaged over all runs (algorithms with a fitness higher than 10000 are not shown), (d-e) show the average progress during the course of a run.

6 Discussion

The results demonstrate that the two performance requirements are fulfilled. The AOSGA can competite with any combination of single operators without a significant decrease in efficiency. The algorithm successfully manages to identify valuable operators during the course of a run and thereby eliminates the need to carry out such experiments by hand.

Concerning the quality of the solutions that the algorithm produces, table 2 and figure 2 indicate that the AOSGA obtains results comparable to the best fixed-op GA. For the two larger problems, the AOSGA converges to these results significantly faster than it's fixed-op counterparts.

As illustrated in figure 3 and 4, operator scheduling definitely takes place, and different operators alternately dominate during a run. Since parameter settings for the individual operators were tuned for single-operator runs, all operators were optimised for all-round best performance. This might not be the optimal setting when the operators are controlled by the AOSGA, which can schedule operators that are specialised for certain tasks (e.g. exploration/exploitation). Introducing task division by specialising operators (e.g. coarse/fine-tuning) the potential of operator scheduling could be exploited to a higher degree.

Futher experimentation should also be conducted concerning the mechanism of adaptation itself. Incorporation of operator probabilities as part of the genotype combined with self-adaptation would require less bookkeeping than the adaptive algorithm presented here. If the performance of this alternative is comparable to the AOSGA, it would therefore be preferable the somewhat cumbersome AOSGA.

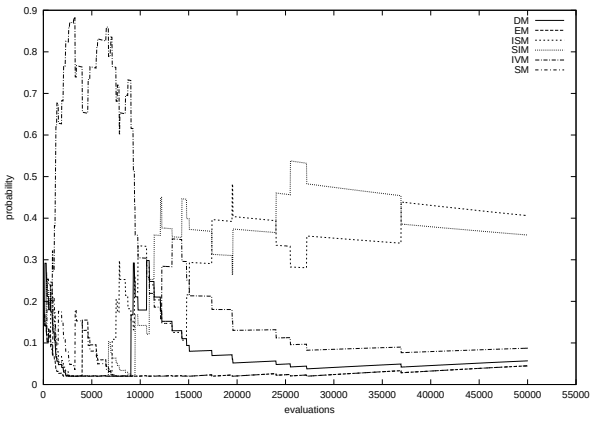


Figure 3: Probability distribution for the mutation operators (gr48)

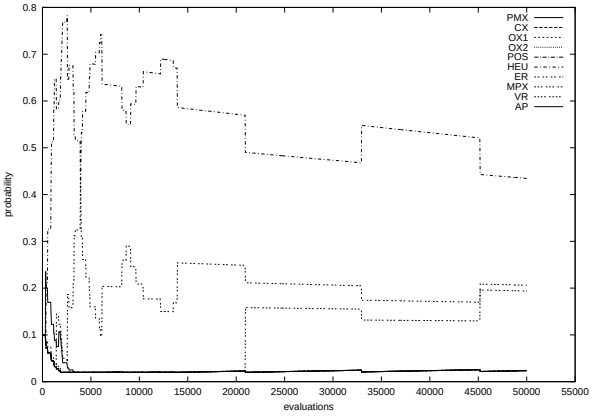


Figure 4: Probability distribution for the crossover operators (gr48)

Acknowledgements

I would like to thank everyone at EVALife for their support and valuable input during this study and the writing of this paper.

Bibliography

[1] L. Davis, "Adapting operator probabilities in genetic algorithms," in *Proceedings of the Third International Conference on Genetic Algorithms* (J. D. Schaffer, ed.), (San Mateo, CA), Morgan Kaufman, 1989.

[2] R. Thomsen and T. Krink, "Self-adaptive operator scheduling using the religion-based ea," in *Proceedings of Parallel Problem Solving from Nature VII (PPSN-2002)*, pp. 214–223, Springer Verlag, 2002.

[3] I. G. S.-K. Agoston E. Eiben and B. A. Thijssen, "Competing crossovers in an adaptive ga framework," in *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, pp. 787–792, 1998.

[4] M. Srinivas and L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 24, no. 4, pp. 656–667, 1994.

[5] B. A. Julstrom, "What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm," in *Proceedings of the Sixth International Conference on Genetic Algorithms* (L. Eshelman, ed.), (San Francisco, CA), pp. 81–87, Morgan Kaufmann, 1995.

[6] A. Tuson and P. Ross, "Cost based operator rate adaption: An investigation," in *Parallel Problem Solving*

from Nature – PPSN IV (H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, eds.), (Berlin), pp. 461–469, Springer, 1996.

[7] D. E. Goldberg and R. Lingle Jr., "Alleles, loci, and the traveling salesman problem," in *Proceedings of the First International Conference on Genetic Algorithms and Their Applications* (J. J. Grefenstette, ed.), Lawrence Erlbaum Associates, Publishers, 1985.

[8] I. M. Oliver, D. J. Smith, and J. R. C. Holland, "A study of permutation crossover operators on the travelling salesman problem," in *Genetic algorithms and their applications : Proc. of the second Int. Conf. on Genetic Algorithms* (J. J. Grefenstette, ed.), (Hillsdale, NJ), pp. 224–230, Lawrence Erlbaum Assoc., 1987.

[9] L. Davis, "Applying adaptive algorithms to epistatic domains," in *Proceedings of the International Joint Conference on Artificial Intelligence*, vol. 1, pp. 161–163, 1985.

[10] G. Syswerda, *Schedule optimization using genetic algorithms*, ch. 21, pp. 332–349. 1991.

[11] J. J. Grefenstette, *Incorporating problem specific knowledge into genetic algorithms*, pp. 42–60. 1987.

[12] D. Whitley, T. Starkweather, and D. Fuquay, "Scheduling problems and traveling salesman: The genetic edge recombination operator," in *Proceedings of the Third International Conference on Genetic Algorithms* (J. D. Schaffer, ed.), (San Mateo, CA), Morgan Kaufman, 1989.

[13] H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer, "Evolution algorithms in combinatorial optimization," *Parallel Computing*, vol. 7, pp. 65–85, 1988.

[14] H. Mühlenbein, "Parallel genetic algorithms, population genetics and combinatorial optimization," in *Proceedings of the Third International Conference on Genetic Algorithms* (J. D. Schaffer, ed.), (San Mateo, CA), Morgan Kaufman, 1989.

[15] M. P. P. Larrañaga, C. M. H Kuijpers and R. H. Murga, "Decomposing bayesian networks: triangulation of the moral graph with genetic algorithms," *Statistics and Computing (UK)*, vol. 7, no. 1, pp. 19–34, 1997.

[16] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin: Springer, 1992.

[17] W. Banzhaf, "The "molecular" traveling salesman," *Biological Cybernetics*, vol. 64, pp. 7–14, 1990.

[18] D. B. Fogel, "An evolutionary approach to the travelling salesman problem," *Biological Cybernetics*, vol. 60, no. 2, pp. 139–144, 1988.

[19] J. H. Holland, *Adpatation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press, 1975.

[20] D. Fogel, "A parallel processing approach to a multiple travelling salesman problem using evolutionary programming," in *Proceedings of the Fourth annual Symposium on Parallel Processing*, (Fullerton, California), pp. 318–326, April 1990.

[21] R. H. M. I. I. P. Larrañaga, C. M. H. Kuijpers and S. Dizdarevic, "Genetic algorithms for the travelling salesman problem: A review of representations and operators," *Artificial Intelligence Review*, vol. 13, no. 2, pp. 129–170, 1999.

[22] Y. Nagata and S. Kobayashi, "Edge assembly crossover: A high-power genetic algorithm for the travelling salesman problem," in *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97)* (T. Bäck, ed.), (San Francisco, CA), Morgan Kaufmann, 1997.

[23] G. Tao and Z. Michalewicz, "Inver-over operator for the TSP," in *Parallel Problem Solving from Nature – PPSN V* (A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, eds.), (Berlin), pp. 803–812, Springer, 1998. Lecture Notes in Computer Science 1498.

[24] H.-K. Tsai, J.-M. Yang, and C.-Y. Kao, "Solving traveling salesman problems by combining global and local search mechanisms," in *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002* (D. B. Fogel, M. A. El-Sharkawi, X. Yao, G. Greenwood, H. Iba, P. Marrow, and M. Shackleton, eds.), pp. 1290–1295, IEEE Press, 2002.

[25] G. Reinelt, "TSPLIB — a traveling salesman problem library," *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–384, 1991.