COMBINATORIAL IMPLEMENTATION OF A PARAMETER-LESS EVOLUTIONARY ALGORITHM

Gregor Papa
Computer Systems Department, Jožef Stefan Institute, Ljubljana, Slovenia

Keywords: Parameter-less, Evolution, Search, Combinatorial, Optimization.

Abstract: The paper presents the combinatorial implementation of an adaptive parameter-less evolutionary-based search. The algorithm is an extension of a basic numerical algorithm that does not need any predefined control parameters values. These values are calculated by the algorithm itself, according to the progress of the search. The efficiency of the proposed autonomous parameter-less algorithm is evaluated by two real-world industrial optimization problems.

1 INTRODUCTION

Finding an appropriate parameter setup of an evolutionary algorithm is a long standing research challenge (Eiben et al., 2007; Kang et al., 2006). Different optimization techniques, like genetic algorithm (GA), differential evolution (DE), evolutionary strategies (ES) or particle swarm optimization (PSO), require several important control parameters that need to be set in advance to ensure effective optimization. The issue of setting the values of various parameters of an evolutionary algorithm is crucial for good performance (Eiben et al., 2007). Furthermore, it has been empirically and theoretically demonstrated, that different values of parameters might be optimal at different stages of the evolutionary process (Stephens et al., 1998; Bäck, 1992). Therefore, particulary interesting are the approaches that would solve any problem without any human intervention for setting the suitable control parameters.

To skip the pre-setting of the control parameters a new technique was developed, which is able to find near-optimal solutions relatively quickly. The presented algorithm is the extension of the original algorithm (Papa, 2008). This version is adapted to combinatorial problems. The algorithm - Parameter-Less Evolutionary Search (PLES) - is based on basic GA, but this algorithm does not need any control parameter, e.g., population size, number of generations, probabilities of crossover and mutation, to be set in advance, but are calculated during the search progress.

2 PARAMETER-LESS SEARCH

The control parameters depend on the behavior and convergence of the found solutions. The pseudo-code of the algorithm is presented in Figure 1. In general, solutions are selected, and recombined in the PLES as they are in the GA, but the implementation of these operators is different. The elitism, selection, and crossover are implemented through forcing-of-better-individuals function, while mutation is split between forcing of better individuals and moving of individuals. The control parameters are never set in advance and are not constant. They are determined each time on the basis of statistical properties of each population. Besides, the algorithm varies the population size. There are some constants used in calculation of the control parameters, but they are treated as meta-parameters. These constants ensure large enough initial population and other parameters in order the algorithm to be operational. Furthermore, these meta-parameters are never changed - not even for different problems.

Setup. The chromosome that represents the solution is constructed upon the number of the variables of the problem, i.e., for the *n* variables the chromosome looks like the string of *n* values.

Initialization. The initial population size ensures large enough population to be able to search within the solution space.

Papa G..
COMBINATORIAL IMPLEMENTATION OF A PARAMETER-LESS EVOLUTIONARY ALGORITHM.
DOI: 10.5220/0003671403070310
In *Proceedings of the International Conference on Evolutionary Computation Theory and Applications* (ECTA-2011), pages 307-310
ISBN: 978-989-8425-83-6

```
Parameter-Less Evolutionary Search
  Set the initial population.
  Evaluate the initial population.
  While stopping criterion not met do.
    Force better individuals to replace worse.
    Move individuals.
    Evaluate the current population.
    Vary population size.
```

Figure 1: The pseudo-code of the PLES algorithm.

$$PopSize_0 = 4\sqrt{n} + 10log_{10}(Range),$$

$$Range = \sum_{j=1}^{n}(max_j - min_j + 1)10^{decplc_j},$$

where n is the number of variables to be optimized, max_j and min_j are the upper and the lower limit of the j-th variable, respectively, and $decplc_j$ is the number of decimal places of the j-th variable. In the case of combinatorial optimization $decplc_j$=0 and $(max_j - min_j + 1)$ is equal to the number of possible combinations of the j-th variable, $combinations_j$, therefore

$$Range = \sum_{j=1}^{n}(combinations_j).$$

Stopping Criterion. The number of overall generations depends on the convergence speed of the best solution found. Optimization proceeds while better solution is found every few generations. But when there is no improvement of the best solution for a $Limit$ ($Limit = 10log_{10}(PopSize_i)$) number of generations, the optimization process stops.

Variable Population Size. During the search process the population size is changed, since the quality of the solution might depend on the size of population. The population size is changed every ($\frac{Limit}{5}$) generations, based on the average change of the standard deviation ($StDev$) of fitness values of solutions over a last few generations.

$$PopSize_{i+1} = \frac{PopSize_i}{\frac{StDev_{i-1}+StDev_{i-2}}{StDev_i+StDev_{i-1}}}$$

The change of population is limited to 20% per change and is further limited to $[\frac{PopSize_0}{5}, 1.1\ PopSize_0]$. The population shrinking enables the search with smaller populations, which is suitable to some types of the problem, or in some stages of the search.

Forcing Better Solution. In every generation worse individuals are replaced with better individuals. After that, every s_{i_j} (variable j of the solution i) is randomly moved up to 20% of the difference between the variable and the limit (upper or lower) of the variable.

$$s_{i_j} = \begin{cases} s_{(i-1)_j} + rnd(max_j - s_{(i-1)_j}) & ;rnd \geq 0 \\ s_{(i-1)_j} + rnd(s_{(i-1)_j} - min_j) & ;rnd < 0 \end{cases}$$

where rnd is a random number [-0.2,0.2].

In the case of combinatorial optimization the s_{i_j} is moved to one of the neighboring values. Here, 20% of positions j are changed to the neighboring value that is defined as 20% of all possible combinations of a variable value. If in $s_{(i-1)}$ the value of the variable j is j_k, then in s_i the variable has the value $j_{(k+rnd\cdot m)}$, where $j = \{j_1, j_2, \ldots j_k, \ldots, j_m\}$; variable j has m possible combinations.

Furthermore, in the case of ordered values the move is performed by switching of randomly chosen pairs of variables inside the solution. Again, up to 20% of variable pairs are switched.

Solution Moving. In the PLES, mutation is realized through the moving of some positions in the chromosome according to different statistical properties. The $Ratio$ of the parameters in the chromosome to be moved is calculated on the basis of standard deviation of the solutions in the previous generations as stated in the following equation.

$$Ratio_i = \frac{StDev_{i-1}}{StDev_{i-3}}n$$

where $StDev_{i-1}$ and $StDev_{i-3}$ are the standard deviation of the solution fitness of the previous generation, and the standard deviation three generations ago, respectively. Here $Ratio_i \in [0 \ldots n]$, and the $Ratio_i$ positions in the chromosome are selected to be moved.

The size of the move is calculated according to the difference between the value of the parameter of the global best solution and current solution, as presented in the following equations

$$MoveRatio = \left(\left|\frac{s_{best_j} - s_{(i-1)_j}}{average_{(i-1)_j} - s_{(i-1)_j}}\right|\right)$$

where s_{i_j} is the value of the parameter j of the current solution i, and s_{best_j} is the value of parameter j of the globally best solution, $average_{(i-1)_j}$ is the average value of the parameter j in the previous generation.

$$Width = s_{best_j} - s_{(i-1)_j}$$

$$s_{i_j} = s_{(i-1)_j} + Direction \times MoveRatio \times Width$$

where *Direction* is randomly selected (-1 or 1).

In the case of combinatorial optimization the s_{i_j} is rounded, since only integer values are accepted.

Solution Evaluation and Statistics. After recombination operations the population is statistically evaluated. Here, the best, the worst, and average fitness value in the generation is found. Furthermore, the standard deviation of fitness values of all solutions in the generation, and the average value of each variable are calculated.

3 INDUSTRIAL PROBLEMS

3.1 Production Planning

Optimal planning of the production in the company that produces different components for domestic appliances has to consider various constraints. The most demanding production stage is the production of cooking hot-plates. The production of the components and parts for all the types of plates is more or less similar, but the standard plate models of the current range differ in size (height, diameter), connector type and power characteristics. Many different models exist due to the various demands of other companies that use those plates for their own cooking appliances. Orders for some particular models usually vary in quantities and deadlines. Orders from the same company are usually also connected with the same deadline. Therefore, their production must be planned very carefully to fulfil all the demands (quantities and deadlines), to maintain the specified amount of different models in the stock, to optimally occupy their workers, and to efficiently use all the production lines. Also, not all the production lines are equal, since each of them can produce only a few different models. The production planning problem consists of finding a production plan that satisfies production time constraints and minimizes production costs (Korošec et al., 2010). The solving involves many specific constraints that need to be considered. The main problem is the exchange delay, caused by adapting production lines to different types of products and supplying the appropriate parts. The manufacturing processes of multiple types of products requires many different steps, and different product parts for completion of each product type. We used the combinatorial version of the PLES algorithm to find the optimal production plan. For the optimal results also some local search procedures were used with the algorithm (Korošec et al., 2010).

The cost function (Eq. (1)), as defined in (Korošec et al., 2010) considers the number of delayed orders (n_{orders}), exchange delay times (t_{exchange}), overall production time (t_{overall}), and the sum of squared days of delayed orders (n_{days}).

$$\begin{aligned} f(P) &= 10^8 \cdot n_{\text{orders}} + 10^4 \cdot t_{\text{exchange}} \\ &\quad + t_{\text{overall}} + n_{\text{days}}. \end{aligned} \tag{1}$$

3.2 Cooling Appliance Setting

Besides the built-in functionalities, the household appliances must have low power consumption energy-efficient optimal-performance. Optimal performance means that the appliance is able to cool to the desired temperature at the lowest possible power consumption. Such optimization usually requires a lot of long-term development measurements or a thorough theoretical analysis of the cooling system and the construction of a complex mathematical model for its simulation (Angeli and Kountouriotis, 2011).

To determine the optimal performance of the refrigerator, we need to perform a set of development measurements for each new type of the appliance (Papa and Mrak, 2010). Thermal processes in cooling systems are by nature very slow. One measurement for determining the power consumption under standard conditions takes several days. To speed-up the development process the temperature simulator was used in connection with the optimizer. The simulation tool simulates temperatures inside the cooling appliance at different modes of regulation. The optimizer uses an evolutionary heuristic search approach to find the optimal set of control parameters iteratively over evolving generations. The mixed combinatorial/numerical version of the PLES algorithm was used to find the optimal control parameters of an appliance, that give an optimal performance with the lowest possible power consumption.

The cost function (Eq. (2)), as defined in (Papa and Mrak, 2010), considers power *consumption* in the calculation interval, the length of the calculation interval, the mean and the reference temperature in C1 and C2 cabinets (*meanT* and *refT*), and the standard deviation (σ_{ON} and σ_{OFF}) of ON and OFF times for the compressor.

$$\begin{aligned} f(P) &= \frac{2800 consumption_{\text{interval}}}{12000 length_{\text{interval}}} \\ &\quad + |meanT_{C1} - refT_{C1}| \\ &\quad + |meanT_{C2} - refT_{C2}| \\ &\quad + \frac{\sigma_{ON}}{1000} + \frac{\sigma_{OFF}}{1000}. \end{aligned} \tag{2}$$

4 RESULTS

The PLES run 20-times for each problem. The experiments were done on 2.2GHz computer, and each run took approximately 5 minutes for production planning, and less than 1 minute for cooling appliance optimization. However, time complexity was not the subject of this evaluation.

4.1 Production Plan

The PLES algorithm was tested on two different real order lists from production company. The Task 1 consists of $n = 711$ orders for 251 products, while the Task 2 consists of $n = 737$ orders for 262 products. In both tasks $m = 5$ production lines are available. The number of evaluations was limited to $500,000$.

While the PLES does not need any parameter, the comparing GA used the following parameters: the population size $N = 100$; the number of generations was $5,000$; the replacement rate $r = 0.2$; the crossover probability $p_\mathrm{c} = 0.7$; the mutation probability $p_\mathrm{m} = 0.005$.

In Table 1 best, mean, worst, and standard deviations of solutions are presented for each task.

Table 1: Results of optimization for Task 1 and Task 2

		PLES	GA
Task 1	Best	1.309×10^8	1.308×10^8
	Mean	1.327×10^8	1.340×10^8
	Worst	1.416×10^8	1.610×10^8
	StD	3.230×10^6	6.526×10^6
Task 2	Best	1.576×10^8	1.611×10^8
	Mean	1.664×10^8	1.748×10^8
	Worst	1.813×10^8	1.914×10^8
	StD	7.390×10^6	7.235×10^6

When comparing with the previous approach of production planning (Korošec et al., 2010), the expert's manual plan for those two tasks had about four-times more delayed orders. This is significantly worse than results obtained by both of the algorithms. Anyway, the PLES algorithm shows faster convergence and proves its search ability to find the solution without the predefined control parameter settings.

4.2 Cooling Appliance

The PLES algorithm was used to speed-up the development process of a new type of a cooling appliance. The PLES was able to find the optimal set of cooling appliance control parameters setting in all runs.

Optimization based on stochastic search method, might give several possible solutions. Criteria for choosing the appropriate one is not only the smallest energy consumption and the desired temperature, but it is also necessary to verify the behavior of the components. Namely, frequent on/off switching of them shortens their life cycle.

5 CONCLUSIONS

The paper presented an adaptive parameter-less evolutionary search for combinatorial problems. The efficiency of the proposed parameter-less algorithm was evaluated by two real-world industrial optimization problems. Following the previously presented numerical results, it was shown, that the combinatorial implementation of the algorithm had faster convergence than comparing algorithm and also it proved its search ability to find the solution without the predefined control parameters. Furthermore, it was shown that the combinatorial implementation of the algorithm is as effective as the numerical one.

REFERENCES

Angeli, D. and Kountouriotis, P.-A. (2011). A stochastic approach to "dynamic-demand" refrigerator control. *Control Systems Technology, IEEE Transactions on*, PP(99):1 –12.

Bäck, T. (1992). The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm. In Männer, R. and Manderick, B., editors, *Proceedings of the 2nd Conference on Parallel Problem Solving from Nature*. North-Holland, Amsterdam.

Eiben, A., Michalewicz, Z., Schoenauer, M., and Smith, J. (2007). Parameter control in evolutionary algorithms. In Lobo, F., Lima, C., and Michalewicz, Z., editors, *Parameter Setting in Evolutionary Algorithms*, volume 54 of *Studies in Computational Intelligence*, pages 19–46. Springer Berlin / Heidelberg.

Kang, Q., Wang, L., and di Wu, Q. (2006). Research on fuzzy adaptive optimization strategy of particle swarm algorithm. *International Journal of Information Technology*, 12(3):65–77.

Korošec, P., Papa, G., and Vukašinović, V. (2010). Application of memetic algorithm in production planning. In *Proc. Bioinspired Optimization Methods and their Applications, BIOMA 2010*, pages 163–175.

Papa, G. (2008). Parameter-less evolutionary search. In *GECCO*, pages 1133–1134.

Papa, G. and Mrak, P. (2010). Optimization of cooling appliance control parameters. In *Proceedings of the 2nd International Conference on Engineering Optimization*, EngOpt2010.

Stephens, C. R., Olmedo, I. G., Vargas, J. M., and Waelbroeck, H. (1998). Self-adaptation in evolving systems. *Artif. Life*, 4:183–201.