Accelerating floating-point to fixed-point data type conversion with evolutionary algorithms

L.S. Rosa✉, C.F.M. Toledo and V. Bonato

The choice of the data type representation has significant impacts on the resource utilisation, maximum clock frequency and power consumption of any hardware design. Although arithmetic hardware units for the fixed-point format can improve performance and reduce energy consumption, the process of tuning the right bit length is known as a time-consuming task, since it is a combinatorial optimisation problem guided by the accumulative arithmetic computation error. A novel evolutionary approach to accelerate the process of converting algorithms from the floating-point to fixed-point format is presented. Results are demonstrated by converting three computing-intensive algorithms from the mobile robotic scenario, where data error accumulated during execution is influenced by external factors, such as sensor noise and navigation environment characteristics. The proposed evolutionary algorithm accelerated the conversion process by up to 2.5× against the state-of-the-art methods, allowing even further bit-length optimisations.

Introduction: Hardware and software optimisations can reduce costs, and improve performance and energy efficiency, significantly. Such improvements are relevant for embedded systems, which are customised for specific applications.

Optimisations related to arithmetic operations play a central role in any customisation process and important project decisions can be made only by knowing the operations type and frequency, data value range and the computation error allowed. Decisions about the most appropriated data type representation and how many bits are necessary for this representation can be made to fulfil the system requirements.

There are different approaches to convert from floating-point to fixed-point format. Most solutions are orientated to digital signal processors applications [1–6]. The main task for converting an algorithm is to estimate the necessary bit lengths for every single variable. This estimation will avoid violating the maximum error defined for the algorithm, which can lead to variables underflow and overflow.

Extensions of bit-length estimation for algorithms with feedback are presented in [7, 8]. The authors in [8] extend [6] to handle unpredictable feedbacks, based on training sets. However, the proposed method is computer intensive and time-consuming for complex algorithms.

We present an evolutionary method to estimate bit lengths for variables with real domain in algorithms according to a maximum error previously defined by the user. The method is validated using classical algorithms for a mobile robotic, where optimisations regarding performance, power consumption and size are considered. The case study is reported covering the EKF-SLAM [9], particle filter (PF) [10] and Gauss-Jordan matrix inversion (MI) [11] algorithms. The main contributions of this Letter are:

- an evolutionary algorithm for the resolution of the bit-length estimation problem;
- a practical solution to accelerate the heavy process of defining fixed-point arithmetic parameters mitigating the whole process of design space exploration in hardware design.

Conversion algorithm: The conversion algorithm estimates the bit range of each fixed-point variable based on an error measure. This error is the difference between the result of the floating-point and the fixed-point execution over a training set β as defined by (1). The parameters $\text{outdata}_{\text{float}}$ and $\text{outdata}_{\text{fixed}}$ are the output of the floating-point and fixed-point versions, respectively:

$$\text{error} = \frac{\text{norm}(\text{outdata}_{\text{float}} - \text{outdata}_{\text{fixed}})}{\text{norm}(\text{outdata}_{\text{float}})} \quad (1)$$

This conversion algorithm is detailed in [6] and contains the following steps: the coarse optimise, which uses a binary search to estimate one single value of precision bit length for all variables of the algorithm and the fine optimise (FO), which makes a variable level optimisation over the coarse optimise results. The optimisation is currently performed using a heuristic guided by the error.

The 'error' calculation is the bottleneck of the conversion algorithm. This occurs because β must be a large set for an adequate estimation and the error estimation has to be made several times in the FO step. In this Letter, the heuristic procedure is replaced by the proposed evolutionary optimise (EO).

Evolutionary optimise: The floating-point to fixed-point conversion problem consists in finding the integer and fractional bit lengths $(m_i.p_i)$ with $i = 1, \ldots, n$, for all n variables of the algorithm to be converted.

From the coarse optimise step previously described, we already have all m_i values estimated and a maximum fractional length p_c for all variables, except for variables representing integer numbers, which have $p_i = 0$ fixed. We define $\boldsymbol{p}$ as a vector of size n, where $p_i \in (0, p_c)$. The vector $\boldsymbol{p}$ will be the chromosome of the EO algorithm and the p_i values, for $i = 1, \ldots, n$, are its alleles.

The proposed EO is able to find solutions where the restriction $E = \text{error} - E_{\max} \leq 0$ is satisfied. The 'error' is the solution error encoded in the $\boldsymbol{p}$ vector and $E_{\max}$ is the maximum acceptable error.

The optimal solution has p_i values near p_c, except by the ones representing integer variables, and most of the variables will have their bit lengths shortly reduced, instead of few variables having a big reduction.

Fitness function and fixed parameters: The problem is a multi-objective optimisation, since we want to reduce the bit lengths and the error. However, the error is a restriction that can be mathematically contoured [12] and reduced to a single objective by applying a penalty/reward according to the error in the fitness function. It is defined the parameter 'severity', which impacts the fitness according to the error as described in the following equation:

$$\text{fitness} = \text{severity} * p_c * 100 * (\text{error} - E_{\max}) + \sum_{i=1}^{n} p_i \quad (2)$$

Since the fitness evaluation implies in calculate E, which is the most time-consuming computation, the proposed EO reduces this computation through the evolution of few individuals over a reduced number of generations. Thus, the following parameters were fixed to help keep the number of fitness evaluations low:

$E\_\max = 1\%$: this value was fixed since it is a fair value for the EKF-SLAM, as presented in [8].
Max_interactions = 100: this parameter limits the number of generations, which is a reasonable value based on empirical tests.
Crossover_rate = 1: the crossover operator is always applied over two parents which will create two children.
Population_size = 100: the population size is limited to 100 individuals, which is also a reasonable value based on previous tests.

Population initialisation: The first step of the EO is to initialise 100 individuals. The value of each allele, except for the ones representing integer variables, is chosen randomly according to an exponential distribution defined in (3), with base = allele_exp and $x_{\max} = p_c$, implying that values near p_c have more probability of being chosen:

$$P(i = x) = \frac{\text{base}^x}{\sum_{k=1}^{x_{\max}} \text{base}^k} \quad (3)$$

Offspring generation: To aim a fast convergence of the EO, it is applied to an elitist gap (μ, λ) [13], with μ = population_size and λ = Gap*μ. The number of individuals for the offspring is defined by (4). Before every generation, the population is sorted by their fitness value in decreasing order:

$$\text{Offspring\_size} = \text{Gap} * \text{population\_size} \quad (4)$$

To generate the offspring population, two individuals are chosen from the sorted population according to an exponential rank with replacement. The rank is given by the position of the individual in the population set, defined by (3), with base = population_exp and $x_{\max}$ = population_size, implying that individuals in the bigger positions have more probability of being chosen to reproduction.

Two children are created from two parents applying a one-point crossover, where a cut-point is a random allele position. Next, each child has a mutation_rate chance of suffering mutation, where one of its values is randomly replaced by a new one according to (3).

Selection and stop conditions: The selection is extremely elitist, where population_size individuals with the best fitness values between the current population and its offsprings are chosen to be in the next generation.

It is expected that the algorithms do not need to run over all max_interactions generations. Thus, we propose a selection intensity metric defined by (5) as another stop criterion. The term $\bar{f}$ is the average of the fitness of all individuals in the population before the offspring generation and selection. The $\overline{f_{sel}}$ is the average of the fitness of the population after the selection. The parameter σ is the covariance of the fitness of all the individuals of the population before the offspring generation and selection

$$\text{Selection\_Intensity} = \frac{|\overline{f_{sel}} - \bar{f}|}{\sigma} \tag{5}$$

Since the selection intensity might be small for the first generations, once the covariance is large for the population close to the initial one, we propose a second stop criterion defined by the takeover rate for half of the individuals in the population.

The takeover rate is calculated in the last stop_variation generations evaluating the average covariance for alleles of all individuals in the 'best half' of the population, after selection, as shown in the following equation:

$$\text{Stable\_Variation} = \overline{\sigma(\text{alleles})} \quad \text{in the last Stop\_Variation generations} \tag{6}$$

Therefore, the EO stops if the best solution has error $\leq E_{max}$, and selection_intensity $\leq$ threshold, and stable_variation $\leq$ threshold. The EO parameters (allele_exp, severity, mutation_rate, population_exp, stop_variation, threshold and gap) were defined by testing their influence in the range proposed in Table 1. We tuned the algorithm with the aim of finding an adequate set of these parameters since they can affect the exploration of the solutions space as well as the convergence speed of the method.

Table 1: Variable parameters of EO, and the range they were varied to analyse the influence of each value on performance of EO, as well as their final value, for EKF-SLAM, PF and MI

Parameter	Range	Final values		
		EKF-SLAM	PF	MI
Allele_exp	1–2	1.3	1.4	1.7
Severity	0–20	4	4	4
Mutation_rate	0–0.02	0.02	0.04	0.01
Population_exp	1–2	1.8	1.9	1.5
Stop_variation	0–1	0.1	0.11	0.14
Threshold	0.05–0.25	0.25	0.25	0.25
Gap	0.1–1	0.1	0.1	0.1

Evolutionary algorithm validation: The EO was executed over 100 different training sets β_j, $j = 1, \ldots, 100$, each one containing 100 EKF-SLAM, PF and MI different executions. The method was adjusted with the parameter values shown in Table 1 for each algorithm.

The solutions were submitted to a quality test. The test consists in running the EKF-SLAM, the PF and MI algorithms over a thousand different executions, counting how many times the error $< E_{max}$ was satisfied, which will define the solution hit_rate. This evaluation was carried out on all training sets β_j for the FO and EO algorithms.

Table 2 summarises the average hit_rate, number of error calculations, the $\sum_{i=1}^{n} p_i$ for each solution given by the EO and by the FO over the training sets β_j and the number of the variables of each algorithm.

Table 2: Number of variables and average Hit_Rate, number of error calculations and $\sum_{i=1}^{n} p_i$ for each solution given by EO and by FO for each respective training set β_j

	Algorithm								
	EKF-SLAM			PF			MI		
Optimise	FO	EO	EO/FO	FO	EO	EO/FO	FO	EO	EO/FO
No. of variables	107	107	1	43	43	1	10	10	1
$\overline{\text{Hit\_rate}}$ (%)	98.3	96.8	1.015	98.6	97.7	1.009	99.2	98.9	1.003
Average no. of error calculations	638	249	2.56	539	221	2.44	220	190	1.16
$\sum_{i=1}^{n} p_i (\times 10^3)$	1.532	0.134	1.46	0.915	0.876	1.05	0.102	0.098	1.04

Discussion and conclusion: In this Letter, we have presented the EO as an optimisation method for bit-length estimation for a floating-point to fixed-point conversion.

Table 2 shows that the EO calculates the error [see (1)], which is the bottleneck of the conversion, achieving 2.56× less for the EKF-SLAM, 2.44× for the FP and 1.16× for the MI. This measure was found not to depend significantly on the training set β used to calculate the error. Thus, if a more robust solution is desired, it is possible to increase the number of elements in β without significant impact on the speed of the EO over the FO algorithm. We emphasise that a single error calculation takes up to 15 min for the EKF-SLAM on a 2.5 GHz Intel Core i5 Processor with 6 Gb RAM.

The MI EO algorithm is 1.16× faster, which is explained by the fact that this algorithm has a smaller number of variables and the EO error calculations increase exponentially with the number of variables of the algorithm. Thus, this indicates that the EO gain over the FO increases with the number of variables in the algorithm.

Table 2 indicates that the EO bit reduction grows with the number of variables, which implies a less robust result, as indicated by the hit_rate. However, the robustness of the EO solutions is not a critical issue once there are hardware-independent solutions that can be applied.

Therefore, the application of the EO saves significant time to estimate the bit lengths of an algorithm, reducing the development time, which is a relevant step to decide which data type is more appropriate to a given design. Furthermore, the reduced bit lengths imply a more compact hardware, with lower energy consumption and a possibly bigger maximum frequency.

28 October 2014
doi: 10.1049/el.2014.3791
L.S. Rosa, C.F.M. Toledo and V. Bonato (*USP, São Paulo, Brazil*)

✉ E-mail: leandrors@usp.br

References

1 Hill, T.: 'AccelDSP synthesis tool floating-point to fixed-point conversion of MATLAB algorithms targeting FPGAs'. White papers, Xilinx, 2006
2 Belanovic, P., and Rupp, M.: 'Automated floating-point to fixed-point conversion with the Fixify environment', *Rapid Syst. Prototyping*, 2005, **28**, pp. 172–178
3 Menard, D., Chillet, D., and Sentieys, O.: 'Floating-to-fixed-point conversion for digital signal processors', *EURASIP J. Appl. Signal Process.*, 2006, **0**, pp. 77–77
4 Shi, C., and Brodersen, R.W.: 'An automated floating-point to fixed-point conversion methodology'. Proc. IEEE Int. Conf. on Acoustical, Speech, and Signal Processing, Hong Kong, April 2003, pp. 529–532
5 Banciu, A., Casseau, E., Menard, D., and Michel, T.: 'Stochastic modeling for floating-point to fixed-point conversion', *IEEE Workshop on Signal Processing Systems*, Beirut, Lebanon, October, 2011, pp. 180–185
6 Roy, S., and Banerjee, P.: 'An algorithm for converting floating-point computations to fixed-point in MATLAB based FPGA design'. Proc. of 41st Annual Design Automation Conf., San Diego, CA, USA, June 2004, pp. 484–487
7 Kinsman, A.B., and Nicolici, N.: 'Automated range and precision bit-width allocation for iterative computations', *Comput.-Aided Des. Integr. Circuits Syst.*, 2011, **30**, (9), pp. 1265–1278
8 Rosa, L.S.O., and Bonato, V.: 'A method to convert floating to fixed-point EKF-SLAM for embedded robotics', *J. Brazilian Comput. Soc.*, 2013, **19**, pp. 181–192
9 Smith, R., Self, M., and Cheeseman, P.: 'Autonomous robot vehicles' (Springer-Verlag New York, Inc., 1990), vol. 64, pp. 167–193
10 Fox, D., Thrun, S., Burgard, W., and Dellaert, F.: 'Particle filters for mobile robot localization' (Springer, New York, 2001), pp. 401–428
11 James, M.L., Smith, G.M., and Wolford, J.C.: 'Applied numerical methods for digital computation' (Harper & Row, New York, 1985), Vol. 2
12 Methods, A., and Vainstein, F.S.: 'Error detection and correction in numerical computations' (Springer, Berlin, Heidelberg, 1991), pp. 456–464
13 Bäck, T., Fogel, D.B., and Michalewicz, Z.: 'Evolutionary computation 1: basic algorithms and operators' (CRC Press, 2000), Vol. 1, doi: 10.1109/SiPS.2011.6088971