# NOVEL ALGORITHMS FOR WORD-LENGTH OPTIMIZATION

*H.-N. Nguyen, D. Menard, O. Sentieys*

IRISA/INRIA, University of Rennes,
6 rue de Kerampont
F-22300 Lannion
{hanguyen,menard,sentieys}@irisa.fr

## ABSTRACT

Digital signal processing applications are specified with floating-point data types but they are usually implemented in embedded systems with fixed-point arithmetic to minimize cost and power consumption. The floating-to-fixed point conversion requires an optimization algorithm to determine a combination of optimum word-length for each operator. This paper proposes new algorithms based on Greedy Randomized Adaptive Search Procedure (GRASP): accuracy-based GRASP and accuracy/cost-based GRASP. Those algorithms are iterative stochastic local searches and result in the best result through many test cases, including IIR, NLMS and FFT filters.

## 1. INTRODUCTION

Most embedded systems integrate digital signal processing applications. These applications are usually designed with high-level description tools to evaluate the application performances with floating-point simulations. Nevertheless, if digital signal processing algorithms are specified and designed with floating-point data types, they are finally implemented into fixed-point architectures to satisfy the cost and power consumption constraints associated with embedded systems. Therefore, the application specification must be converted into fixed-point. The aim of the conversion process is to determine, for each data, the number of bits allocated to the integer and fractional parts. This process consists of two main steps corresponding to the determination of the position of the binary point and the optimization of the data word-length. In the first step, the data dynamic is determined analytically by interval arithmetic or affine arithmetic, or by simulation-based methods. This step ensures that there is no overflow.

In the second step, the fractional word-length is determined. An optimization procedure searches for a combination of optimum word-length for each operator. The optimization process is performed under constraint, which ensures that the numerical accuracy is sufficient to maintain the application performance. The optimization problem is combinatorial and can be defined as follow:

$$\min \mathscr{C}(\mathbf{b}) \quad \text{subject to} \quad \lambda(\mathbf{b}) \geq \lambda_{\min} \tag{1}$$

where $\boldsymbol{b}$ is the vector representing the combination word-length of each operator, $\mathscr{C}$ is the cost function that has to be minimized, $\lambda$ is the numerical accuracy and $\lambda_{\min}$ is the minimal accuracy that can accepted. Generally, $\mathscr{C}$ corresponds to power, energy consumption, area or delay; $\lambda$ in many applications is the signal-to-quantization noise ratio. There is no relation in literature between the nature of $\mathscr{C}$ and the optimization algorithm choice. In architecture having fine-grained word-length granularity, most of the word-length in a given range is supported. This granularity is obtained when the architecture can be configured or designed at the bit level like in ASIC or in FPGA when logic elements are used.

When the word-length vector size $|\boldsymbol{b}|$ is large, the optimization time increases significantly. There are many signal grouping techniques to reduce this size, but they also reduces the flexibility in word-length choices. In this work, medium and large-size optimization problems are considered. Deterministic approaches based on greedy algorithms [1, 2, 3, 4] allow obtaining a solution quickly, but the quality of this solution decreases when the number of variables increases. Stochastic approaches like genetic algorithm [5] improve the quality of the solution but require a high optimization time.

In this paper, two new algorithms for the word-length optimization procedure, based on GRASP, are proposed. Compared to existing methods, our proposition yields better results and has a complexity between deterministic methods and stochastic methods. The paper is organized as follows. Some notions in application of combinatorial optimization in word-length determination are given in Section 2. Previous works in word-length optimization algorithms are presented in Section 3. Our algorithms are detailed in Section 4. Simulation results are discussed in Section 5 and we conclude our work in Section 6.

## 2. DEFINITIONS

In this section, some definitions to facilitate the presentation in the paper are given. The minimum word-length of an operator is its smallest word-length so that the accuracy criterion (1) is satisfied if other operators are at their largest supported word-length. The *minimum word-length set* (MWS) is a vector $\boldsymbol{b}^{\min} = (b_1^{\min}, b_2^{\min}, ..., b_N^{\min})$ where each $b_k^{\min}$ is the minimum word-length of $k^{\text{th}}$ operator.

The minimum uniform word-length, noted as $b^{\text{uni}}$, is the smallest value satisfying $\lambda(\boldsymbol{b}^{\text{uni}}) \geq \lambda_{\min}$ with $\boldsymbol{b}^{\text{uni}} = (b^{\text{uni}}, b^{\text{uni}}, ..., b^{\text{uni}})$.

## 3. RELATED WORK

The word-length optimization problem is NP-complete so there is no practical method able to find an optimal solution. Instead, algorithms attempt to build an acceptable solution with different heuristics. It is not an easy task to classify or to compare different algorithms. In this paper, algorithms are divided by their nature: deterministic or non-deterministic.

### 3.1 Deterministic algorithms

K.I. Kum and W. Sung use an *exhaustive search* and a *heuristic procedure* in [1]. The exhaustive search starts from the MWS, evaluates all word-length combinations having a distance of one bit. If no solution is found, the distance is increased until a solution is found. It is not a complete search because there is no guarantee that a solution at distance $d$ is always better, in term of cost, than every solution at distance $d+1$. Then once a solution at distance $d$ is found, the algorithm ignores all other solutions at distance $d+1$ and above, and may miss the best solution. The heuristic strategy is by some mean a minor modification of the exhaustive search. Instead of increasing the distance, in each step it increases the word-length of every operator by one bit to quickly find a solution. Then, in the refinement process, it tries to reduce the word-length of each operator while still keeping the accuracy criterion satisfied. In both algorithms, the solution is not always optimal, even not sub-optimal.

A *sequence search*, or *min+1 bit*, is presented in [2]. It is a greedy algorithm, where an operator is chosen in each step to increase its word-length by 1 bit. The criterion used in those choices is the accuracy: each operator is temporarily increased by 1 bit, the one that results in the best accuracy is selected. There are other choices, for example the discrete gradient.

Constantinides et al. propose a *max–1 bit* greedy algorithm in [3]. The authors consider that the cost function and the accuracy function are not increasing with the word-length set, but suppose that they are increasing with the uniform word-length (mono-variable). The starting point for greedy descent is $k\boldsymbol{b}^{\text{uni}}$, where $k > 1$ is a scaling factor. The criterion used to choose the best operator in each word-length descent step is the cost reduction.

Another greedy algorithm is presented in [4]. Instead of using the complexity (cost) or distortion (accuracy), the authors use a complexity–distortion measure, which is a linear combination of them. While it is not trivial to find the good weighting coefficient $\alpha_k$, the usage of normalized values of complexity and distortion measure with a coefficient $\alpha_k$ equal to 0.5 leads to the best solution in an IIR design.

*Branch and Bound* algorithms are also used in word-length optimization. As this is a complete search, it is only practical for small problems. In [6], some improvements are proposed. The authors suggest that the search space is only between $\boldsymbol{b}^{\min}$ and $\boldsymbol{b}^{\text{uni}}$. Also, if a node $\boldsymbol{b}^t$ does not satisfy the accuracy criterion, all nodes $\boldsymbol{b}^k$ where $b_n^k \leq b_n^t \quad \forall n = 1..N$ are ignored. In [7] other improvements are presented. It consists in operator reordering by accuracy sensitivity, pre-eliminate bad branches and use limited number of word-length values for each operator. Those values are around the result found in a relaxed optimization problem where word-lengths can be non-integer. However, there is no proof that the optimum in $\mathbb{Z}^+$ is near an optimal solution in $\mathbb{R}^+$.

*Mixed Integer Linear Programming* (MILP) is used in [8] to achieve the global optimum. However, because of very long optimization time in a moderate-size problem, this approach is usually used to assert other algorithms performance in not very complex problems.

### 3.2 Stochastic algorithms

*Simulated Annealing* (SA) is an iterative optimization where each iteration tries to improve the solution. It may step back to a worse solution with a specific probability. In [9], the word-length optimization problem is transformed to MILP and solved by SA. In [10], an Adaptive Simulated Annealing (ASA) is used to optimize a PID controller in $\mathbb{R}^+$ then the word-lengths are approximated from that result.

[11] proposes an application of ASA with MWS as the starting point. For a polynomial approximation, the solutions are very close to the global optima. However, the authors also remark that for more complex designs, the search space is huge and the optimization time will be long.

In [12], a *Genetic Algorithm* (GA) is used to optimize LMS filter coefficients and the results are compared to random search. In [13], a hybrid GA associated with a gradient-based greedy search is proposed. This new approach leads to a faster convergence speed in comparison with the GA.

There are many attempts to use *Multi-Objective GA* (MOGA) in word-length optimization. The weighted-sum method is used to find solution for the FFT coefficients in an MC-CDMA receiver [14]. In [15], the weighted-sum technique is again used to optimize the word-length in two different applications, a FIR and a DCT. However, in those works, there is no comparison with deterministic algorithms. In [5], multi-objective GA is compared to greedy-like algorithms and known to have better performance in small problems: 4-variable IIR and 7-variable IIR.

## 4. PROPOSED ALGORITHMS

In this section, firstly a better greedy algorithm, known as *Tabu Search* or *steepest-descent–mildest-ascent* is proposed for word-length optimization. Then, this algorithm serves the local search step in GRASP. As we will see later, the criterion for local search could be the accuracy or the accuracy/cost ratio. Therefore, we propose two corresponding GRASP algorithms: accuracy-based GRASP (GRASP-a) and accuracy/cost-based GRASP (GRASP-ac).

### 4.1 Tabu Search

Currently all greedy algorithms used in word-length optimization are mono-directional: it is either steepest-descent (*max–1*) or mildest-ascent (*min+1*). The proposed algorithm, based on Tabu Search [16], allows movements in both directions. The set $T$ containing *tabu operators* is used to avoid useless or infinite loops.

The procedure is presented in Algorithm 1. In the first part, at lines 5–15, the gradient associated to each operator is calculated. If an operator reaches its word-length limit, it is added into the tabu list. Next, at lines 19–30, depending on the direction, the operator with either the largest or the smallest gradient is selected in this step. If this selection makes the current accuracy go below minimum allowable value, the direction is reversed.

It can be proven that the performance of this algorithm is at least as the same as greedy search.

#### Criterion for direction search

At each step of the algorithm, the word-length of one operator is modified to move towards the final solution. A criterion has to be defined to select the best direction *i.e.* the operator for which the word-length has to be modified. The criterion is based on the computation of the discrete gradient of the cost and the accuracy. Two criteria for selecting the best direction have been considered. The first criterion computes

**Algorithm 1** Tabu search in word-length optimization

**Require:** solution $\boldsymbol{b}$
**Ensure:** better solution than $\boldsymbol{b}$
1: $T \leftarrow \emptyset$ *tabu operators*
2: bestCost $\leftarrow$ NaN; bestWL $\leftarrow \emptyset$
3: direction $\leftarrow \lambda(\boldsymbol{b}) \geq \lambda_{\min} ? -1 : 1$ *determine direction*
4: **while** $|T| < N$ **do**
5: **for all** $1 \leq k \notin T \leq N$ **do** *calculate criterion*
6: **if** cannot move $\boldsymbol{b}_k$ **then**
7: $T \leftarrow T \cup \{k\}$
8: **else**
9: $\boldsymbol{b}^{\text{next}}, k \leftarrow$ next position of $\boldsymbol{b}$ at operator $k$
10: $\nabla_k \leftarrow f_{\text{dir}}\left(\boldsymbol{b}^{\text{next},k}, \boldsymbol{b}\right)$
11: **if** $\lambda(\boldsymbol{b}^{\text{next}}, k) \geq \lambda_{\min}$ **then**
12: update bestCost, bestWL if necessary
13: **end if**
14: **end if**
15: **end for**
16: **if** $|T| = N$ **then**
17: stop
18: **end if**
19: **if** direction $> 0$ **then**
20: $j \leftarrow \arg\max \nabla_k$; $b_j \leftarrow b_j + 1$
21: **if** $\lambda(\boldsymbol{b}) \geq \lambda_{\min}$ **then**
22: direction $\leftarrow -1$; $T \leftarrow T \cup \{j\}$
23: **end if**
24: **else**
25: $j \leftarrow \arg\min \nabla_k$
26: $b_j \leftarrow b_j - 1$
27: **if** $\lambda(\boldsymbol{b}) < \lambda_{\min}$ **then**
28: direction $\leftarrow 1$
29: **end if**
30: **end if**
31: **end while**
32: **return** bestWL

the gradient on the accuracy and corresponds to the one used in the well-known *min+1 bit*

$$\nabla_{k/\lambda} = f_{\text{dir}}\left(\boldsymbol{b}^{\text{next},k}, \boldsymbol{b}\right) = \frac{\lambda(\boldsymbol{b}^{\text{next},k}) - \lambda(\boldsymbol{b})}{d(\boldsymbol{b}^{\text{next},k}, \boldsymbol{b})} \tag{2}$$

with $\boldsymbol{b} = (b_1, ..., b_k, ...b_N)$, $\boldsymbol{b}^{\text{next},k} = (b_1, ..., \text{next}(b_k), ...b_N)$ and $d$ is a distance metric. The term $\text{next}(b_k)$ represents the next value of $b_k$ and is equal to $b_k + 1$ if the difference between two available word-lengths is of 1 bit and the direction is positive. Within the same condition, the distance between two vectors $d(\boldsymbol{b}^{\text{next},k}, \boldsymbol{b}) = 1$.

Amongst deterministic algorithms, *min+1 bit* does not always give a good result. It takes sometimes the wrong direction and returns very bad results. To improve this criteria, *the cost and the accuracy* are taken into account as follows

$$\nabla_{k/\lambda\mathscr{C}} = \frac{\nabla_{k/\lambda}}{\nabla_{k/\mathscr{C}}} = \frac{\lambda(\boldsymbol{b}^{\text{next},k}) - \lambda(\boldsymbol{b})}{\mathscr{C}(\boldsymbol{b}^{\text{next},k}) - \mathscr{C}(\boldsymbol{b})} \tag{3}$$

This criterion selects the direction which minimizes the cost increase and maximizes the accuracy increase.

### 4.2 Greedy Randomized Adaptive Search Procedure (GRASP)

Deterministic greedy algorithms, although simple, can not improve existing solution even if they have more time. In the latter case, there are many works on stochastic local search. We propose here a GRASP-based algorithm for word-length optimization. This algorithm, as illustrated in Algorithm 2, is a multi-start process consisting of two phases: construction phase and local search phase. In the construction phase, a randomized search algorithm is used to find a sub-optimal solution $\boldsymbol{b}^g$ and then from this solution $\boldsymbol{b}^g$ a local search is applied to refine the solution. The randomize aspects allow avoiding local minima. For the local search phase, the tabu search algorithm presented in Section 4.1 is used.

**Algorithm 2** Pseudo-code for GRASP procedure

1: $\boldsymbol{b} \leftarrow \boldsymbol{b}^{\min}$
2: **while** maximum number of iterations not reached **do**
3: $\boldsymbol{b}^g \leftarrow$ greedy randomized solution
4: Local search from $\boldsymbol{b}^g$
5: $\boldsymbol{b} \leftarrow$ which is better$(\boldsymbol{b}, \boldsymbol{b}^g)$ *update the best solution*
6: **end while**

#### Construction phase

The construction phase is presented in Algorithm 3. In this phase, a greedy randomized search is used to get a random sub-optimal solution. From line 3–5, the quality of each candidate is calculated. A parameter $T_{\text{RCL}}$ that is the size of the Restricted Candidate List (RCL) is defined. The RCL contains the top $T_{\text{RCL}}$ candidates at each step. A candidate from this list is selected with a uniform probability $1/T_{\text{RCL}}$ and is used for the next iteration. The parameter $T_{\text{RCL}}$ determines thus the variant of solutions in construction phase. The usual greedy algorithm is a special case where $T_{\text{RCL}} = 1$.

**Algorithm 3** GRASP: Solution construction phase

1: $\boldsymbol{b} \leftarrow \boldsymbol{b}^{\min}$
2: **while** $\lambda(\boldsymbol{b}) < \lambda_{\min}$ **do**
3: **for** $k = 1$ to $N$ **do**
4: $\nabla_k \leftarrow f_{\text{dir}}\left(\boldsymbol{b}^{\text{next},k}, \boldsymbol{b}\right)$
5: **end for**
6: RCL $\leftarrow$ top $T_{\text{RCL}}$ candidates with best $q_k$
7: $i \leftarrow$ random value in RCL
8: $\boldsymbol{b} \leftarrow \text{next}(\boldsymbol{b}, i)$
9: **end while**

## 5. SIMULATIONS AND RESULTS

In this section, we show that both GRASP-a and GRASP-ac have better results than deterministic algorithms and genetic algorithms. Energy consumption is used as the cost function.

### 5.1 Accuracy-based GRASP (GRASP-a)

In order to assert the performance of our algorithm, we compare it with other well-known algorithms. Amongst deterministic algorithms, a *min+1 bit* greedy search and CDM[4] are selected. CDM is a greedy algorithm using a linear combination of normalized quality and cost function, that usually

has better results than the basic *min+1*. Amongst stochastic algorithms, a MOGA [5] is used in this comparison.

Our test cases include 32-, 64- and 128-point Fast Fourier Transforms (FFT), a 128-point Normalized Least Mean Square (NLMS) filter and a cascaded canonic biquad Infinite Impulse Response (IIR) filter. Each application can have different operator assignations, therefore many test cases and many problem sizes can be experimented.

The purpose of our test bench is to compare different algorithm performance. The reduction of the problem size by signal grouping or the optimization/high-level synthesis coupling[17, 18] is not in the scope of this paper. Medium-size problems are considered, because small problems do not well differentiate algorithm performance. Large problems result in very bad deterministic algorithm performance, also require much more simulation time in MOGA to have an adequate solution. In all of our tests, MOGA have a population size of 90 and stop after 500 generations, where there are still dominated individuals. This size is good for small and average size problems, but not good enough for larger ones. However, with this size, MOGA already has the longest optimization time. In GRASP-a, the parameter $T_{\text{RCL}}$ is fixed to 3, and the algorithm stops after 10 iterations.

The optimization problem is presented in (1), where we search for an optimal cost $\min \mathscr{C}(\mathbf{b})$ subject to an accuracy not less than $\lambda_{\min}$. The optimization results are presented in Table 1 for a 64-point FFT with 12 variables, in Table 2 for the NLMS filter with 25 variables, and in Figure 1 for the IIR filters with 14, 18 and 36 variables. $n_{\mathscr{C}}$ and $n_{\lambda}$ are the number of function evaluations. All results are energy expressed in Joule. To obtain those results, each deterministic algorithm runs once and each stochastic algorithm runs 20 times to get the average result. It is noteworthy that while deterministic algorithms do not have stable performance, stochastic algorithms like MOGA and GRASP always return acceptable solution. Our proposed algorithm always performs better than the others.

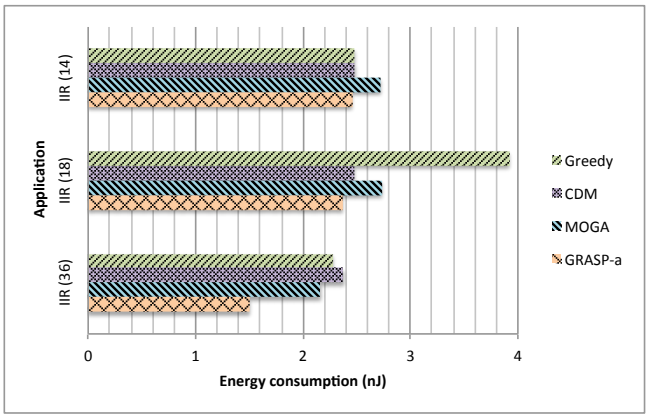
GRASP-a performs significantly better than MOGA in term of result quality and optimization time for small and average size problems.

Table 1: Performance comparison: FFT-64 (12 variables)

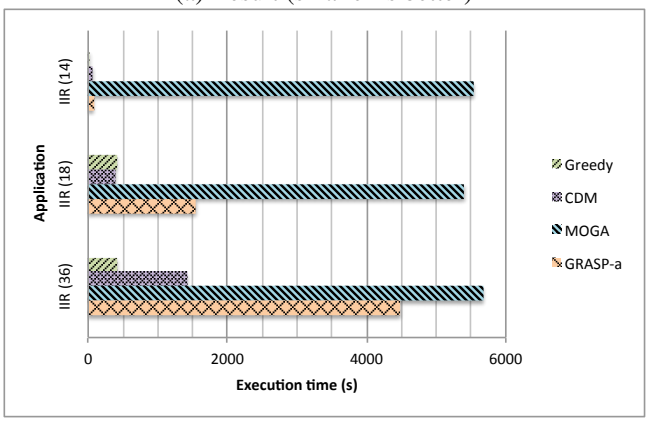
| Algorithm | Time (s) | $n_{\mathscr{C}}$ | $n_{\lambda}$ | Result (J) |
|---|---|---|---|---|
| Greedy [2] | 67 | 1 | 1731 | $33.19 \times 10^{-8}$ |
| CDM [4] | 126 | 1623 | 1867 | $22.15 \times 10^{-8}$ |
| MOGA [5] | 2715 | 40213 | 41412 | $6.030 \times 10^{-8}$ |
| GRASP-a | 278 | 165 | 7985 | $5.213 \times 10^{-8}$ |

Table 2: Performance comparison: NLMS-128 (25 variables)

| Algorithm | Time (s) | $n_{\mathscr{C}}$ | $n_{\lambda}$ | Result (J) |
|---|---|---|---|---|
| Greedy | 47 | 1 | 426 | $2.150 \times 10^{-8}$ |
| CDM | 55 | 277 | 439 | $2.150 \times 10^{-8}$ |
| MOGA | 6001 | 42084 | 42234 | $2.513 \times 10^{-8}$ |
| GRASP-a | 414 | 395 | 3630 | $2.127 \times 10^{-8}$ |




(a) Result (smaller is better)




(b) Time (smaller is better)

Figure 1: Solution quality and optimization time for different IIR structures: 14 variables, 18 variables and 36 variables.

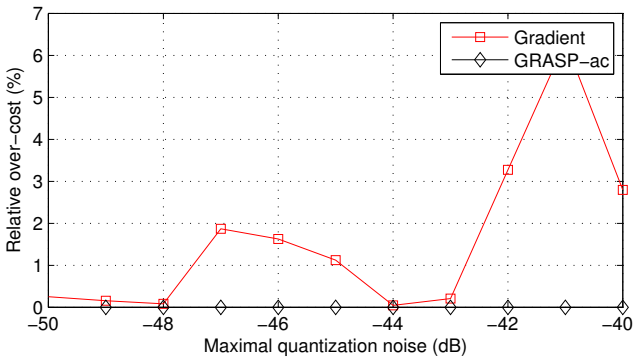
### 5.2 Accuracy/Cost-based GRASP

In this section, the performance of GRASP-ac is measured against the gradient descent search in FFT applications, with a 64-point FFT (12 variables) and a 128-point FFT (28 variables). In each FFT, we test the algorithms in 11 values of minimal accepted accuracy. To demonstrate the performance of GRASP-ac against the gradient search, only 5 iterations are used instead of 10 in GRASP-a. The results are presented in Figure 2. GRASP-ac performs at least as well as gradient descent, and is up to 6% and 10% better with a reasonable execution time.

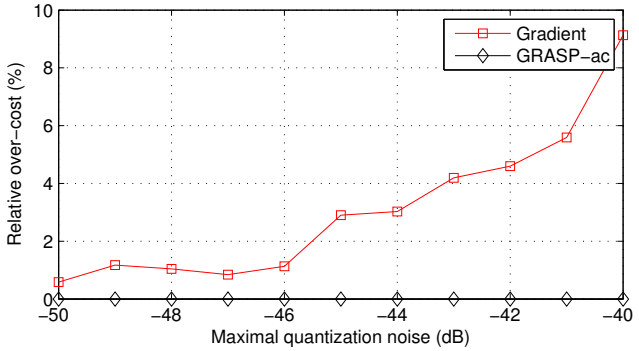
### 5.3 Further remarks

Finally, the combination of GRASP-a and GRASP-ac is compared against greedy search and gradient descent to show how it improves deterministic algorithm performance. All algorithms are tested in IIR, FFT and NLMS applications each having different numbers of variables and different minimal accepted accuracy. Overall, 189 optimization problems are created and tested with different algorithms. The most improved solutions with our GRASP are presented in Figure 3. The solution quality is normalized: the best solution is equal to 100%. Even gradient descent gives quite good solution, though GRASP could improve the result by 10–20%.

The experiment have been carried out on a mono-core processor. Given that GRASP is a multi-start process, it can




(a) 64-point FFT (12 variables)




(b) 128-point FFT (28 variables)

Figure 2: Comparison between the gradient descent and GRASP-ac on FFT applications. Figures represent the relative over-cost compared to the best solutions.

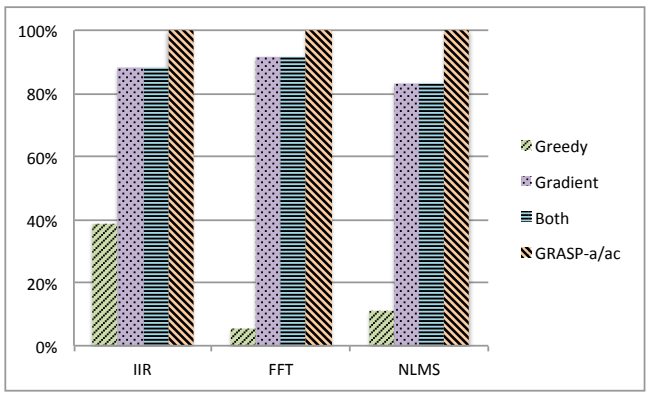



Figure 3: Solution quality of optimization algorithms: the combined GRASP-a/ac and the others. Results are normalized: 100% is the best solution.

benefit easily from multi-core processor and the execution time can be reduced significantly.

## 6. CONCLUSION

In this paper, a new stochastic local search algorithm is proposed, in coupled with a tabu search. Our experiments in different medium-size optimization problems show that this algorithm has very good performance in comparison to other deterministic and stochastic algorithms. One of the main disadvantages of deterministic algorithm is that the performance can not be improved if more time is available. GRASP does not have this problem. Moreover, compared to Genetic Algorithms, the complexity of GRASP is moderate and is easily adjustable. In the future, different selection criteria in the greedy randomized search and intelligently-adjustable RCL will be considered. Moreover, the appropriate number of iterations will be experimented. The proposed algorithm would also be tested with larger problems.

## REFERENCES

[1] W. Sung and K. Kum, "Simulation-based word-length optimization method for fixed-point digital signal processing systems," *IEEE Trans. Signal Process.*, vol. 43, no. 12, pp. 3087–3090, 1995.

[2] K. Han, I. Eo, K. Kim, and H. Cho, "Numerical word-length optimization for CDMA demodulator," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 4, Sydney, May 2001, pp. 290–293.

[3] G. Constantinides, P. Cheung, and W. Luk, "The multiple wordlength paradigm," in *Proc. IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, Rohnert Park, CA, 2001, pp. 51–60.

[4] K. Han and B. L. Evans, "Optimum wordlength search using sensitivity information," *EURASIP Journal on Applied Signal Processing*, pp. 1–14, 2006.

[5] K. Han, "Automating transformations from floating-point to fixed-point for implementing digital signal processing algorithms," Ph.D. dissertation, University of Texas at Austin, 2006.

[6] H. Choi and W. Burleson, "Search-based wordlength optimization for VLSI/DSP synthesis," in *Proc. IEEE Workshop on VLSI Signal Processing (VLSI–SP)*, San Diego, CA, Oct. 1994, pp. 198–207.

[7] D. Menard, D. Chillet, and O. Sentieys, "Floating-to-Fixed-Point Conversion for Digital Signal Processors," *EURASIP Journal on Applied Signal Processing*, vol. 14, 2006.

[8] G. Constantinides, P. Cheung, and W. Luk, *Synthesis and optimization of DSP algorithms*. Kluwer Academic Publishers, 2004.

[9] F. Catthoor, H. De Man, and J. Vandewalle, "Simulated-annealing-based optimization of coefficient and data word-lengths in digital filters," *International Journal of Circuit Theory and Applications*, vol. 16, no. 4, 1988.

[10] S. Chen, J. Wu, R. Istepanian, and J. Chu, "Optimizing stability bounds of finite-precision PID controller structures," *IEEE Trans. Autom. Control*, vol. 44, no. 11, pp. 2149–2153, 2002.

[11] D. Lee, A. Gaffar, R. Cheung, O. Mencer, W. Luk, and G. Constantinides, "Accuracy-guaranteed bit-width optimization," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 10, pp. 1990–2000, 2006.

[12] R. Nambiar, C. Tang, and P. Mars, "Genetic and learning automata algorithms for adaptive digital filters," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, San Francisco, CA, Apr. 1992, pp. 41–44.

[13] S. Ng, S. Leung, C. Chung, A. Luk, and W. Lau, "The genetic search approach. A new learning algorithm for adaptive IIR filtering," *IEEE Signal Process. Mag.*, vol. 13, no. 6, pp. 38–46, 1996.

[14] N. Sulaiman and T. Arslan, "A multi-objective genetic algorithm for on-chip real-time optimisation of word length and power consumption in a pipelined FFT processor targeting a MC-CDMA receiver," in *Proc. NASA/DoD Conference on Evolvable Hardware (EH)*, Washington, DC, 2005, pp. 154–159.

[15] A. Ahmadi and M. Zwolinski, "Word-Length Oriented Multiobjective Optimization of Area and Power Consumption in DSP Algorithm Implementation," in *Proc. International Conference on Microelectronics (MIEL)*, Belgrade, 2006, pp. 614–617.

[16] F. Glover, "Tabu Search – Part I," *INFORMS Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.

[17] N. Herve, D. Menard, and O. Sentieys, "Data wordlength optimization for FPGA synthesis," in *Proc. IEEE Workshop on Signal Processing Systems (SiPS)*, Athens, Nov. 2005, pp. 623–628.

[18] B. Le Gal, A. Caaliph, and E. Casseau, "Bit-Width Aware High-Level Synthesis for Digital Signal Processing Systems," in *IEEE International SoC Conference (SOCC)*, Austin, TX, Sep. 2006, pp. 175–178.