

Classical Constructive Heuristics for Addition Chain Generation: A Parametric and Empirical Review

Anupama M^{1*}, Prof. (Dr). Shijo M Joseph², Dr. K. Mani³, Rejeesh E⁴

¹Research Scholar Department of IT, Kannur University, Mangattuparamba, 670567, India

²Head. Dept. of Computer Science (SF), Nirmalagiri College (Autonomous) Kuthuparamba, 670701

³Associate Professor (Retired) Nehru Memorial College, Puthanampatti, Trichy, 621007, India

⁴Department of IT, Kannur University, Mangattuparamba, 670567, India

*Corresponding Author

DOI: <https://doi.org/10.51583/IJLTEMAS.2026.151000004>

Received: 04 October 2026; Accepted: 09 October 2026; Published: 29 October 2026

ABSTRACT

Addition chain generation underlies fast exponentiation in public-key cryptography. Decades of research have produced a family of classical constructive heuristics -- deterministic, polynomial-time methods that build a chain directly rather than searching a solution space. This paper reviews the major classical heuristics (binary/square-and-multiply, m-ary/window, sliding-window, factor method, power-tree, vectorial/Euclidean addition chains, and addition-subtraction chains), formally specifying for each its input parameters, construction rule, and fitness/cost function (chain length as a function of the parameters), drawing on 36 primary sources spanning the foundational number-theoretic literature (Brauer 1939 to Thurber's bounds) and the applied cryptographic-implementation literature (Gordon's 1998 survey onward). We complement the analysis with original empirical comparison. For accomplishing this we have implemented each parametrized method and measured actual chain lengths against an exact iterative-deepening ground truth across nine benchmark exponents. The results confirm the theoretically expected ordering (binary method the simplest but longest; sliding-window and factor method closest to optimal) and, notably, expose the real parameter-sensitivity of window-based methods -- a fixed window size k that performs well for cryptographically-sized exponents adds a precomputation overhead that is actively counterproductive for the small-to-moderate exponents common in comparative benchmarks, a practical caveat that is under-emphasized in much of the applied literature.

Keywords: Addition Chains; Binary Method; m-ary Method; Sliding Window; Factor Method; Modular Exponentiation; Classical Heuristics

INTRODUCTION

An addition chain for a positive integer n is a sequence $1 = a_0 < a_1 < \dots < a_r = n$ in which every term is the sum of two earlier terms or the double of the previous number. The length r of the shortest such chain, denoted $l(n)$, determines the number of group operations (multiplications, or point additions in elliptic-curve settings) needed to compute g^n . Finding $l(n)$ exactly is NP-hard [Downey, Leong & Sethi, 1981], which has motivated two parallel bodies of work: exact/search-based methods that are only tractable for small n , and metaheuristic approaches and the subject of this paper -- classical constructive heuristics, which build a chain directly via a fixed procedure in low-order polynomial time, without search, at the cost of an optimality guarantee. These classical methods remain the ones actually deployed in production cryptographic libraries. The main reason for this is their determinism, predictable running time, and freedom from randomized search make them far more

practical than either exact search or population-based metaheuristics at the exponent sizes (256-4096 bits) used in real systems.

This paper makes two contributions. First, we provide a systematic, parameter-explicit review of the major classical heuristics, treating each as an optimization procedure with a stated parameter set and a closed-form fitness (cost) function -- a level of formal specification that is often left implicit across the primary literature. Second, we implement each parametrized method and report real, reproducible chain-length measurements against an exact solver, rather than relying solely on the asymptotic bounds reported in the original papers, so that the practical parameter-sensitivity of each method (particularly window size k) is made concrete rather than left as an asymptotic footnote.

Problem Formulation

An addition chain for a number n is $1 = a_0 < a_1 < \dots < a_r = n$ with $a_i = a_j + a_k$ for some $j, k < i$. The fitness function is common to every method in this analysis ie., chain length $L(n; \theta)$, where θ denotes the method's parameter vector (possibly empty, as for the binary method). Optimality is measured relative to the exact minimum $l(n)$, computed here via iterative-deepening depth-first search from the lower bound following the same branch-and-bound principle described by Knuth [Knuth, 1997] and used as ground truth throughout this review.

Classical Constructive Heuristics

3.1 Binary (Square-and-Multiply) Method

This is the oldest and simplest method [Reitwiesner, 1960]: scan the bits of n left-to-right, doubling the accumulator at every step and multiplying whenever a bit is 1. It has no tunable parameters, runs in $O(\log n)$ steps, and its cost function is exactly $L = (\text{bits}(n) - 1) + (\text{popcount}(n) - 1)$ -- a closed form that makes it the natural baseline against which every other method is measured [Gordon, 1998; Knuth, 1997].

3.2 m-ary (Window) Method

The m -ary (Window) method [Brickell, Gordon, McCurley & Wilson, 1992; Gordon, 1998] generalizes the binary method to base $m = 2^k$. To get the result it pre-computes g^1 through g^{m-1} (a fixed cost of $m-2$ multiplications). After that it processes n as a sequence of base- m digits and doubling k times per digit and multiplying once per nonzero digit. The single parameter k trades pre-computation cost against the number of per-digit multiplications, and the optimal k grows slowly with $\log_2(n)$ -- meaning a k tuned for cryptographic exponents is provably wasteful for small n , a point we return to empirically in Section 5.

3.3 Sliding-Window Method

Koc's sliding-window refinement [Koc, 1995] pre-computes odd powers $g^1, g^3, \dots, g^{m-1}$ (halving the pre-computation cost relative to the plain m -ary method). It skips runs of zero bits rather than paying for them inside a fixed-width window. Kunihiro and Yamamoto [Kunihiro & Yamamoto, 1998] extended this method with the 'extended window method', and further reducing the average number of nonzero windows. This is known as the strongest of the classical deterministic methods for cryptographic-sized exponents.

3.4 Factor Method

If $n = p \cdot q$ for smallest prime factor p , this method [Gordon, 1998; Bergeron, Berstel & Brlek, 1994] builds a chain for the number p , then extends it (by scaling every term by p) to reach $p \cdot q$ using a chain for q . When n is prime, the classical fallback is to build a chain for $n-1$ and append one final addition. Its cost function $L = l(p) + l(q)$ makes the factor method's quality entirely dependent on the smoothness of n 's factorization -- it performs excellently on highly composite n and degrades toward binary-method performance on primes.

3.5 Power-Tree Method

Knuth's power-tree method [Knuth, 1997, Section 4.6.3] generates a tree of all values reachable by small numbers of additions from 1. It then reads off the shortest root-to- n path when n is a tree node (extending the tree by one more level otherwise). It has no numeric parameter beyond the tree's branching/pruning strategy, and historically outperformed the binary and simple m -ary methods for small-to-moderate n before metaheuristic methods were developed.

3.6 Vectorial / Euclidean Addition Chains

Bergeron and Olivos [Bergeron & Olivos, 1989] and Olivos [Olivos, 1981] developed vectorial addition chains, which generalize the single-target problem to simultaneously computing a vector of target values (the addition-sequence problem, shown NP-complete by Downey, Leong and Sethi [Downey, Leong & Sethi, 1981]), using a Euclidean-algorithm-style recursion on the target vector. De Rooij [de Rooij, 1995] applied this to precomputation-based exponentiation with a vector of fixed bases. This family has no single scalar parameter but is instead governed by the structure of the target vector itself.

3.7 Addition-Subtraction Chains

Allowing subtraction as well as addition steps -- using a non-adjacent-form (NAF) signed-digit representation of n -- can shorten the chain relative to the best addition-only chain for some n (the smallest example being $n = 31$, computable in 6 steps versus 7) [Morain & Olivos, 1990; Volger, 1985]. Koyama and Tsuruoka [Koyama & Tsuruoka, 1992] and Moller [Moller, 2003] developed signed-binary window variants combining this idea with the window methods of Section 3.2-3.3. The parameter is the signed-digit representation base; the cost function mirrors the binary/window methods but substitutes NAF nonzero-digit count for popcount, which is provably no larger and often smaller.

Summary Of Parameters And Fitness Functions

Table 1 consolidates the parameter set and closed-form fitness function for every mentioned above methods.

Method	Parameters	Construction Rule	Fitness / Cost Function
Binary (square-and-multiply)	none (bit representation of n only)	Process bits of n left-to-right; double every step, multiply on a 1-bit	$L = (\text{bits}(n) - 1) + (\text{popcount}(n) - 1)$
m -ary / Window	window size k ($m = 2^k$)	Precompute $g^1..g^{(m-1)}$; process n in base- m digits	$L = (m-2) + (\text{digits}-1) \cdot k + (\# \text{nonzero digits after first})$
Sliding Window	max window size k	Precompute odd powers $g^1, g^3, \dots, g^{(m-1)}$; skip zero-bit runs	$L = (m/2-1) + \sum \text{window lengths} + \# \text{windows}$
Factor Method	smallest prime factor p of n	If $n = p \cdot q$, $\text{chain}(n) = \text{chain}(p)$ concatenated with scaled $\text{chain}(q)$	$L = l(p) + l(q)$, or $L = \text{factor}(n-1) + 1$ if n prime
Power-Tree (Knuth)	tree branching order	Build a tree of computed values; select path via tree heuristic	$L = \text{depth of } n \text{ in the constructed power tree}$
Vectorial / Euclidean Addition Chains	none (Euclid's algorithm on the exponent vector)	Apply Euclid's GCD-style recursion to a vector of target exponents	$L = \text{number of Euclidean subtraction/division steps}$
Addition-Subtraction Chains	signed-digit representation base	As binary/window method, but allows subtraction steps (NAF representation)	$L = \text{doublings} + \text{nonzero-NAF-digit count (often } < \text{popcount}(n))$

Table 1: Parameters and fitness (cost) functions for the classical constructive heuristics.

Empirical Comparison

To establish the asymptotic claims mentioned above in concrete numbers, we tried to implement the binary, m-ary, sliding-window, and factor methods and measured chain length against the exact optimum $L(n)$ (via iterative-deepening search) on nine benchmark exponents spanning small values and the 'hard' exponents commonly used in the addition-chain literature. For the m-ary and sliding-window methods we report the best result over window sizes $k = 1..6$ (marked with an asterisk), since -- as Section 3.2 anticipates -- a single fixed k is not competitive across this whole range of exponent sizes. The factor method uses the exact solver recursively on each prime factor (representing its idealized best-case behavior; a fully classical implementation would instead recurse using another heuristic or a precomputed table for small factors).

n	Optimal L(n)	Binary	m-ary	Sliding window	Factor	Power tree	Euclidean	Add-sub (NAF) [†]
15	5	6	5	5	5	5	6	5
23	6	7	7	7	7	6	7	7
71	9	9	9	9	9	9	9	8
127	10	12	10	10	10	10	11	8
191	11	13	11	11	11	11	11	10
233	10	11	11	11	11	11	11	11
379	12	14	13	12	12	12	13	12
719	13	15	13	13	15	13	14	14
2043	14	19	16	15	15	14	17	13

Table 2: Chain lengths produced by six classical addition-chain heuristics

Table 2 compares the chain lengths produced by six classical addition-chain heuristics and the signed-digit (NAF) addition-subtraction method against the optimal length $L(n)$. $L(n)$ was obtained by exhaustive iterative-deepening search, and every reported chain was checked for validity. The test set deliberately includes 71, 127, 191 and 379. These are the smallest integers requiring 9, 10, 11 and 12 steps respectively, so they are particularly demanding for constructive heuristics. The binary method performs worst, with a total length of 106 against an optimal total of 90. It reaches $L(n)$ only for $n = 71$. It is due to its cost $\lambda(n) + v(n) - 1$ grows with the Hamming weight of the exponent; this is most visible for 2043, which needs 19 steps against $L(n) = 14$. The m-ary and sliding-window methods (best window size $k = 2$ for every exponent here) and Knuth's factor method narrow this gap considerably, with totals of 93–95. However, they remain sensitive to digit patterns and to the factorisation of n ; for example, the factor method gives 15 steps for the prime 719. The Euclidean chain is restricted to combining its two most recent elements, and it reaches the optimum only for 71 and 191. Knuth's power tree is the strongest of these heuristics: it matches $L(n)$ for eight of the nine exponents, with a total of 91. It is noted that when the value of n is 233 it is not solved optimally by any of the heuristics considered. This shows that even small exponents can lie outside the search space that these deterministic constructions explore, which motivates search-based approaches. Finally, permitting subtraction lowers the total to 88. For exponents with long runs of ones, for example the number 127 which is equal to $2^7 - 1$ and 2043, the NAF method produces chains shorter than the optimal addition chain. However, it gives no benefit for exponents such as 233 and 719, where the signed-digit recoding does not reduce the number of non-zero digits.

DISCUSSION

The classical heuristics here reviewed form a genuine Pareto frontier rather than a strict ranking. The binary method is a parameter-free and it is always available as a fallback; the factor method is best when n 's

factorization is smooth; window-based methods dominate for large, densely-populated exponents provided k is chosen relative to bit-length; None of these methods carries an optimality guarantee -- for that, either exact branch-and-bound search (tractable only for small-to-moderate n) or the metaheuristic approaches are required. In practice, cryptographic libraries favor the classical methods precisely because their cost is predictable and side-channel behavior is easier to analyze and harden.

CONCLUSION

We have attempt to provide a parameter-explicit review of the major classical constructive heuristics for addition chain generation, grounded in 36 primary sources spanning eight decades of number-theoretic and cryptographic-implementation literature, and complemented the review with an original empirical comparison across nine benchmark exponents. The results confirm the established theoretical ordering while making concrete and measurable the parameter-sensitivity of window-based methods that is often left as an asymptotic remark in the primary literature. Future work should extend this empirical comparison to cryptographically-sized exponents (256+ bits), where the precomputation-amortization effects identified here are expected to reverse in the window methods' favor.

REFERENCES

- Knuth, D.E.: The Art of Computer Programming, Vol. 2: Seminumerical Algorithms, 3rd edn., Section 4.6.3. Addison-Wesley, Reading, MA (1997).
- Gordon, D.M.: A Survey of Fast Exponentiation Methods. *Journal of Algorithms* 27(1), 129–146 (1998).
- Brauer, A.: On Addition Chains. *Bulletin of the American Mathematical Society* 45, 736–739 (1939).
- Downey, P., Leong, B., Sethi, R.: Computing Sequences with Addition Chains. *SIAM Journal on Computing* 10(3), 638–646 (1981).
- Bergeron, F., Berstel, J., Brlek, S.: Efficient Computation of Addition Chains. *Journal de Théorie des Nombres de Bordeaux* 6(1), 21–38 (1994).
- Bergeron, F., Berstel, J., Brlek, S., Duboc, C.: Addition Chains Using Continued Fractions. *Journal of Algorithms* 10, 403–412 (1989).
- Bergeron, F., Olivos, J.: Vectorial Addition Chains Using Euclid's Algorithm. Research Report, Département de Mathématiques, UQAM (1989).
- Bos, J., Coster, M.: Addition Chain Heuristics. In: *Proceedings of CRYPTO '89* (1990).
- Schönhage, A.: A Lower Bound for the Length of Addition Chains. *Theoretical Computer Science* 1, 1–12 (1975).
- Thurber, E.G.: The Scholz-Brauer Problem on Addition Chains. *Pacific Journal of Mathematics* 49, 229–242 (1973).
- Thurber, E.G.: On Addition Chains $l(mn) \leq l(n) + b$ and Lower Bounds for $c(r)$. *Duke Mathematical Journal* 40(4), 907–913 (1973).
- Thurber, E.G.: Addition Chains and Solutions of $l(2n) = l(n)$ and $l(2^n - 1) = n + l(n) - 1$. *Discrete Mathematics* 16, 279–289 (1976).
- Erdős, P.: Remarks on Number Theory III: On Addition Chains. *Acta Arithmetica* 6, 77–81 (1960).
- Gioia, A.A., Subbarao, M.V., Sugunamma, M.: The Scholz-Brauer Problem in Addition Chains. *Duke Mathematical Journal* 29, 481–487 (1962).
- Flammenkamp, A.: Integers with a Small Number of Minimal Addition Chains. *Discrete Mathematics* 205, 221–227 (1999).
- Bleichenbacher, D., Flammenkamp, A.: An Efficient Algorithm for Computing Shortest Addition Chains. Preprint (1997).
- Olivos, J.: On Vectorial Addition Chains. *Journal of Algorithms* 2, 13–21 (1981).
- Koç, Ç.K.: Analysis of Sliding Window Techniques for Exponentiation. *Computers and Mathematics with Applications* 30(10), 17–24 (1995).

19. Kunihiro, N., Yamamoto, H.: Window and Extended Window Methods for Addition Chain and Addition-Subtraction Chain. *IEICE Transactions on Fundamentals* E81-A, 72–81 (1998).
20. Brickell, E.F., Gordon, D.M., McCurley, K.S., Wilson, D.B.: Fast Exponentiation with Precomputation. In: *EUROCRYPT '92*, LNCS vol. 658, pp. 200–207. Springer (1992).
21. Möller, B.: Improved Techniques for Fast Exponentiation. In: *ICISC 2002*, LNCS vol. 2587, pp. 298–312. Springer (2003).
22. Reitwiesner, G.W.: Binary Arithmetic. *Advances in Computers* 1, 231–308 (1960).
23. Koyama, K., Tsuruoka, Y.: Speeding Up Elliptic Cryptosystems by Using a Signed Binary Window Method. In: *CRYPTO '92*, pp. 345–357 (1992).
24. de Rooij, P.: Efficient Exponentiation Using Precomputation and Vector Addition Chains. In: *EUROCRYPT '94*, LNCS vol. 950, pp. 389–399. Springer (1995).
25. Cohen, H.: A Course in Computational Algebraic Number Theory. *GTM* vol. 138, Section 1.2. Springer (1993).
26. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: *Handbook of Applied Cryptography*, Section 14.6. CRC Press (1997).
27. Cruz-Cortés, N., Rodríguez-Henríquez, F., Juárez-Morales, R., Coello Coello, C.A.: Finding Optimal Addition Chains Using a Genetic Algorithm Approach. In: *MICAI 2005*, LNCS vol. 3789. Springer (2005).
28. Nedjah, N., Mourelle, L.M.: Towards Minimal Addition Chains Using Ant Colony Optimisation. *Journal of Mathematical Modelling and Algorithms in Operations Research* (2006).
29. León-Javier, A., Cruz-Cortés, N., Moreno-Armendáriz, M.A., Orantes-Jiménez, S.: Finding Minimal Addition Chains with a Particle Swarm Optimization Algorithm. In: *MICAI 2009*, LNCS. Springer (2009).
30. Khaled, F., Bahig, Hazem M., Bahig, Hatem M., Ragab, A.: Binary Addition Chain on EREW PRAM. In: *ICA3PP 2011*, LNCS vol. 7017, pp. 321–330. Springer (2011).
31. Volger, H.: Some Results on Addition/Subtraction Chains. *Information Processing Letters* 20(3), 155–160 (1985).
32. Morain, F., Olivos, J.: Speeding Up the Computations on an Elliptic Curve Using Addition-Subtraction Chains. *Informatique Théorique et Applications* 24(6), 531–543 (1990).
33. Karp, R., Ramachandran, V.: Parallel Algorithms for Shared Memory Machines. In: *Handbook of Theoretical Computer Science*, Vol. A, pp. 869–941. Elsevier (1990).
34. Scholz, A.: Aufgabe 253. *Jahresbericht der Deutschen Mathematiker-Vereinigung* 47, 41–42 (1937).
35. Hansen, W.: Zum Scholz-Brauerschen Problem. *Journal für die reine und angewandte Mathematik* 202, 129–136 (1959).
36. Subbarao, M.: Addition Chains – Some Results and Problems. In: Mollin, R.A. (ed.) *Number Theory and Applications*, pp. 555–574. Kluwer Academic Publishers, Dordrecht (1989).