UDC 004.02
IRSTI 20.53.15

LOCAL SEARCH IN NON-DETERMINISTIC FINITE AUTOMATA WITH EXTENSIONS

Mirzakhmet Syzdykov
al-Farabi Kazakh National University, Almaty, Kazakhstan
mspmail598@gmail.com
ORCID ID: https://orcid.org/0000-0002-8086-775X

Abstract. In this work we present the theoretical approach over solving the back-reference problem in regular expression matching within the almost polynomial time using local search within the memory, while within the growth of capturing groups we obtain the exponential results: for this purpose we develop the modified matching algorithm operating on non-deterministic finite automata within the modified search algorithm and presence of the specific method also over extended regular expressions. This is made due to the algorithm which can be adjusted for approximate searching allowing us to imply extended operators and features of modern regular expressions like intersection, subtraction and complement, as well as back-references. The review of past work on this issues is also done: to the present time there is no discrete algorithm in systems like automata for local search. Thus, we obtain the new result of matching the pattern locally while the simulating algorithm works as usual. The obtained result also refers to the membership problem with local bound which can be set in the main algorithm presented in this article.

Keywords: Back-reference, Finite Automata, Local Search, Extended regular expressions.

Introduction

As of the regular expression languages nature and the proposed solution of extending it with a repeater like back-reference, in the modern era this feature is used in analyzing specific sequences in the medicine [1].

Thus, the back-reference problem is considered widely as NP-complete and the backtracking algorithm is used for the matching problem [2].

However, due to the memoization [2] the backtracking process can be optimized — as we use the same approach in giving the memoization of the matching process for back-references and corresponding capturing groups.

The solution also covers the regular expression features like look-ahead and extended operators.

Typically we define the non-deterministic finite automaton (NFA) as a tuple of alphabet A, states S, list of transitions T, starting state s0 and list of finishing or accepting states:

$$<A, S, T, s_0, F> \quad (1)$$

In this work we will present the modified NFA for organizing the local search on back-reference, positive or negative look-ahead and look-behind and extended set of operators like intersection, complement and subtraction.

Related past work review

In [1] the importance and application of back-reference operator is given.

In [2] the memoization principle is covered which we use in simulating the NFA within the capturing groups and back-references on the counter state – this method is presented further in this article.

The work [3] deals with the backtracking matchers as they are commonly used in matching regular expression with back-references as an easy alternative to the approach presented in this paper, which uses the memoization of the state context and counter simulation for the state referred as ending in NFA for back-reference operator in regular expression.

In [4] the subset construction of converting NFA to deterministic finite automaton (DFA) is given.

In [5] the construction of NFA for the regular expression is given.

In [6] the past work for converting regular expression to finite automata is presented.

In [7] the definition of transitional NFA is given with respect to the context of captured groups to be

optimized towards memory consumption.

In [8] the NFA constructions are presented for extended operators in regular expression like intersection, subtraction and complement which are to be used in local search during matching process as well as a back-reference operator.

The modern work [9] describes the logical constructions to model the features like positive or negative look-ahead and look-behind which is met in Perl and other regular expression flavors.

In [10] the algorithm for converting regular expression directly into deterministic finite automaton is presented. It's important to us as the example of the past work.

In [11] the effective polynomial algorithm is given for extended regular expressions. We will show further that our automaton is as well quadratic in memory consumption as in complexity.

In [12] the theory of complexity is presented – this is a seminal work of theory of complexity when the optimization problem is to be non-reliable if we deal with solution set explosion along the algorithm to check the correct solution in this set.

We will state further that according to the NP-completeness of back-reference problem [13], using the counter heuristics we achieve polynomial results in time and memory.

The work [14] considers the state explosion problem when converting NFA to DFA using techniques to avoid it. We consider modified or extended NFA in this work which can be built by any existing algorithm and mainly is from linear up to quadratic number of states in it.

The work [15] is important for us in the sense of exact polynomial complexity, the algorithm for Traveling Salesman Problem (TSP) given in the article is exponential in time and can be considered as not optimal in the sense of modern computer architecture.

Methodology

Before proceeding to the NFA, we can mention that it can be transformed to DFA according to the algorithm by subset construction [4]. We can define the Thompson NFA [5] for our algorithm as it's not a conceptual decision and can be changed even to the derivative constructions by Berry-Sethi [10] or other algorithms [6]. The use of NFA instead of DFA for local search isn't principal as it will be shown further.

The preliminary and the key idea is to define the context of captured groups and back-references in regular expression as following set of positions in the input string for each of the states in NFA:

$$Context(S) = \{ (L_i, R_i), (B_j, E_j) \}, i = 1..N, j = 1..M, \quad (2)$$

where the N is the number of capturing groups, M is the number of back-references and the values in the set L and R are the positions of matched captures for the corresponding back-reference in the searched text – B and E.

To use the memoization principle we will extend the particular polynomial matching algorithm or even family of such algorithms [5] with the context principle where each context is dependent on the input string position is stored in each reached state during the simulation, while the modern solution can solve in general more effectively as to the memory consumption [7].

Thus, we obtain the polynomial results for our algorithm according to our results, when its complexity depends on the state of context in the state of simulated NFA as the stored set (2) is saved in memory – thus giving the possibility to exclude repeated intervals in the input during the pattern matching process.

In this work for simulation purposes of local search we present the NFA with counter edges, where the counter is decreased by one at each step of the matching process, when the position in input is increased by one, and is included in the set of active states when the configuration complies to the present input, as well, as initialized to the counter value of the matched sub-string and is included for candidate when the counter value equals zero.

The counter for the set of states S in automaton can be defined as follows:

$$Counter(S) = \{ (S_i, C_i) \}, C_i \text{ in } N+, \quad (3)$$

where Si is the state of non-deterministic automaton and Ci is the corresponding counter value.

Thus, the local search even for extended constructions [8] can be achieved by implying the counters in (3) and simulating the NFA within the active zero value when the state can be used further in the non-local search before it will be pushed to the structure like stack which is typically used in the simulation of NFA along the input string. The back-references and extended operators [8] are to be matched locally with respect to the counter and state composition (3) when it's pushed to the stack for processing with the newly acquired counter value and, when, it equals to zero within the process of matching simulation, it becomes active or popped out from the same stack and forwarded to the matching procedure.

We obtain the following algorithm of simulating the local search with delay using the counter values (3):

Algorithm-1. The simulation of NFA with the local delay using counter.

```
define stack T;

define set of states S;

define the input string P;

T := {};

S := { Starting state of automaton };

for i in 1..|P| loop

for s in S loop

        if matched(P[i..j]): T.push ((s, |P[i..j]|));

        S.pop (s);

end loop

for t in T loop

        t.C := t.C – 1;

        if t.C = 0 then

                T.pop (t);

                S := S + { t.S };

        end if

end loop

end loop
```

The above algorithm procedure works in at most quadratic time and memory as it's bound to all the sub-strings from the input.

The Algorithm-1 can be extended for the back-references in “matched” function when the context (2) is used to compare the captured group and the current input for equivalence after which the state is to be considered.

Finally we state that the local search works on the extended NFA within the sub-program Algorithm-1 as follows:

$$<A, S, T, s_0, F, Context(S), Counter(S)> \quad (4)$$

The automaton (4) can be reduced only to counters if the back-references aren’t present.

Thus, we obtain the principally new results for simulating NFA for extended operators like intersection subtraction and complement [8] as well as for a back-reference within the space of counter states and local search using structures in (2) and (3). These structures are to be used for simulating NFA for effective memoization of the captured groups and back-references (2) and for pop-operation delay, according to the matched string during local search using the structure (3).

As back-reference problem is considered to be NP-complete – we solve it in polynomial time using the extra memory consumption using memoization principle and delay processing.

The class of NP is to be considered exponential or even factorial. The backtracking algorithm for matching without complexity optimization gives almost factorial time, while in our method the memoization and counter state space is used which is limited to the quadratic of input string as all the sub-strings are to be considered for matching.

In [8] using the algorithm from [4] the construction of deterministic finite automata (DFA) is presented. However, due to the state explosion problem these DFA cannot be universal and are bound to special cases where the explosion occurs. In this work we give the definition of local search with structures (2) and (3) which can be used in NFA which is linear in size to the regular expression with features like extended operators (intersection, complement and subtraction) and back-references.

The features like pre-matching [9] can be also modeled in NFA instead of building the DFA with re-written structure using &-operator [8].

In general the local search is also effective as other algorithms [11] when computed according to the specific cases.

So as to NP-completeness as we use quadratic memory in pattern matching routine the following statement of the complexity classes holds true:

$$P = NP: M \text{ in } P^2, \quad (5)$$

where M is the memory factor.

Conclusion

Despite the fact that our idea is based upon the memory consumption and the complexity reduction to the polynomial order, there are open questions concerning the determinism and the configuration of reached states, which can change the behaviour of the matching process. This question remains open, however, we can be sure that local search with counter heuristics solves the problem of in-memory matching.

The other point of view is the equality of P and NP classes with the use of memory (5) in computations for the NP-complete back-reference problem.

References

[1] Hassan, Mohammed Ahmed Ezzelregal, and Mohamed Elsayed Hasan. "Finding a Tandem Repeats Motifs in the Completed Genomes of Human Coronavirus (hku1) Which is Identified as a Hotspot Region for the Viruses Recombination by Using Regular Expression Language." (2019).

[2] van der Merwe, Brink, et al. "Memoized Regular Expressions." *International Conference on*

Implementation and Application of Automata. Springer, Cham, 2021.

[3] Berglund, Martin, Frank Drewes, and Brink Van Der Merwe. "Analyzing catastrophic backtracking behavior in practical regular expression matching." *arXiv preprint* arXiv:1405.5599 (2014).

[4] Rabin, Michael O., and Dana Scott. "Finite automata and their decision problems." *IBM journal of research and development* 3.2 (1959): 114-125.

[5] Thompson, Ken. "Programming techniques: Regular expression search algorithm." *Communications of the ACM* 11.6 (1968): 419-422.

[6] Champarnaud, J-M., J-L. Ponty, and Djelloul Ziadi. "From regular expressions to finite automata." *International journal of computer mathematics* 72.4 (1999): 415-431.

[7] Laurikari, Ville. "NFAs with tagged transitions, their conversion to deterministic automata and application to regular expressions." *Proceedings Seventh International Symposium on String Processing and Information Retrieval*. SPIRE 2000. IEEE, 2000.

[8] Syzdykov, Mirzakhmet. "Deterministic automata for extended regular expressions." *Open Computer Science* 7.1 (2017): 24-28

[9] Berglund, Martin, Brink van der Merwe, and Steyn van Litsenborgh. "Regular Expressions with Lookahead." *Journal of Universal Computer Science* 27.4 (2021): 324-340.

[10] Berry, Gerard, and Ravi Sethi. "From regular expressions to deterministic automata." *Theoretical computer science* 48 (1986): 117-126.

[11] Roşu, Grigore. "An effective algorithm for the membership problem for extended regular expressions." *International Conference on Foundations of Software Science and Computational Structures.* Springer, Berlin, Heidelberg, 2007.

[12] Cook, Stephen. "The P versus NP problem." *The millennium prize problems* (2006): 87-104.

[13] Alevoor, Praveen, Pratik Sarda, and Kalpesh Kapoor. "On Decidability and Matching Issues for Regex Languages." *Proceedings of International Conference on Advances in Computing*. Springer, New Delhi, 2013.

[14] Yu, Xiaodong, Bill Lin, and Michela Becchi. "Revisiting state blow-up: Automatically building augmented-fa while preserving functional equivalence." *IEEE Journal on Selected Areas in Communications* 32.10 (2014): 1822-1833.

[15] Held, Michael, and Richard M. Karp. "A dynamic programming approach to sequencing problems." *Journal of the Society for Industrial and Applied mathematics* 10.1 (1962): 196-210.