VarMatch: A fast, parallel, and memory-efficient method for the variant matching problem

Chen Sun and Paul Medvedev

The Pennsylvania State University, USA

1 Introduction

Small variant ($\leq$ 30bp) calling is widely used in medical and genetic research to identify how genome mutations are related to phenotypes of interest. Variant matching is the problem of comparing different sets of variant calls, to determine the variants that are in common between the sets or unique to each set. Variant matching can be done to (1) compare the performance of different tools with respect to each other or with respect to a ground truth. (2) extract high-confidence variants for an individual by taking the intersection of calls from multiple callers, and (3) find variants that are shared or unique across different individuals.

A set of small variants is typically represented as a collection of VCF entries, where each entry contains a position of the reference genome and the alternate diploid allele (e.g. sequence) in the donor. The most straightforward variant matching algorithm is to directly match identical VCF entries. However, it can fail to match two different VCF entries that nevertheless result in the same diploid donor genome. Normalization and decomposition [1–3] have been used to alleviate these problems, however, there are still alternate representations for the same variant that are not matched [4]. An alternate approach is to formulate and solve an appropriate optimization problem that finds, roughly speaking, the largest number of matches [4]. This method can detect equivalent variants unmatched by heuristic algorithms, but still suffers from large memory usage.

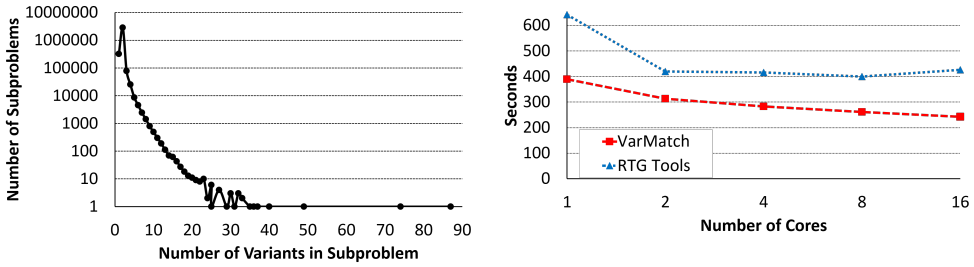
An additional limitation is that these approaches can only support maximizing the number of total matched VCF entries. However, this is sensitive to whether a tool represents complex variants as a single entry or as multiple, decomposed, entries. A more representation-invariable optimization criteria would be to maximize the number of matched nucleotides. In other cases, such as comparing multiple callers to a ground truth set, it is desirable to instead maximize the total number of matched entries from the ground truth set only.

2 Summary

To address these problems, we introduce a new algorithm VarMatch. VarMatch is an exact algorithm for variant matching that is guaranteed to find matching variants under a wide variety of optimization criteria. VarMatch employs a provably optimal divide and conquer strategy to partition the set of variants into disjoint subproblems. Because each subproblem is typically very small, we can use an exact dynamic programming algorithm similar to [4] (for the maximum number of matches optimization criteria) or even brute force (for other criteria) to solve each subproblem. While our algorithm has exponential running time in the worst case, we demonstrate it performs very fast in practice and uses an order of magnitude less memory than [4]. This can be crucial for applications in medical settings, where the software may be run on embedded processors or portable devices. VarMatch is also a parallel algorithm that scales over multiple processors and/or threads. Additionally, our divide and conquer strategy makes it easy to support any optimization criteria for doing matches, since even a brute force implementation is practical for the small subproblems. We have implemented several scoring functions in VarMatch: (1) maximize the total number of matched entries, (2) maximize the number of matched entries from one of the call sets, and (3) maximize the total number of matched bases. VarMatch is implemented as a user-friendly software package that will be available on GitHub if accepted.

3 Results

We consider the variant matching problem, roughly defined as follows: given a pair of variant sets $\langle \mathcal{V}, \mathcal{W} \rangle$, find subsets $V \in \mathcal{W}$ and $W \in \mathcal{W}$ such that applying V and W results in the same diploid sequence, and $f(V, W)$ is maximized. The function f can be almost any computable function, with the most natural



(a) (b)

Fig. 1: Effectiveness of problem partitioning (a) and parallelization (b) of VarMatch

one $f(V, W) = |V| + |W|$. If we consider a reference genome interval without any variants, we can split the input variants into those to the left and to the right of the interval. Our main theoretical result states that, given a sufficiently long and non-repetitive interval, the solution to the variant matching problem on $\langle \mathcal{V}, \mathcal{W} \rangle$ is equivalent to the union of the solutions to the problem on $\langle \mathcal{V}_{\text{left}}, \mathcal{W}_{\text{left}} \rangle$ and $\langle \mathcal{V}_{\text{right}}, \mathcal{W}_{\text{right}} \rangle$. This theorem is significant for two reasons. First, it leads to an exact, parallel, fast and low-memory divide-and-conquer algorithm that partitions the large problem into smaller subproblems which can be solved with a brute-force algorithm. Second, it allows the use of any reasonable optimization criteria, which other algorithms do not allow.

Table 1 illustrates evaluation of VarMatch on two published real data sets [2] with single thread, comparing the accuracy, memory usage(RAM) and running time of VarMatch to the normalization approach (based on Vt [1] followed by direct matching) and to RTG Tools [4]. For dataset CHM1 (Table 1a), we take variant call sets on the same sequencing data of the CHM1hTERT cell line. Variants were called separately by FreeBayes and HaplotypeCaller of GATK. For dataset NA12878 (Table 1b), we take variant call sets by Platypus and UnifiedGenotyper of GATK on NA12878 cell line. Both RTG Tools and VarMatch match more VCF entries than Vt at the cost of more resources, but VarMatch uses less running time and an order of magnitude less memory than RTG Tools.

Method	Matched Entries		RAM (Gb)	Time (s)
	FB	HC		
Vt	2,778,372	2,778,372	0.004	216
RTG Tools	2,843,004	2,911,802	48	642
VarMatch	2,843,004	2,911,802	4.7	389

(a) Dataset CHM1. Variants are called by Freebayes(FB) and HaplotypeCaller of GATK(HC).

Method	Matched Entries		RAM (Gb)	Time (s)
	PT	UG		
Vt	4,072,823	4,072,823	0.004	258
RTG Tools	4,228,302	4,414,044	34	836
VarMatch	4,228,302	4,414,044	5.5	704

(b) Dataset NA12878. Variants are called by Platypus(PT) and UnifiedGenotyper of GATK(UG).

Table 1: Comparison of VarMatch to two other variant matching methods on different datasets.

Figure 1a shows the effectiveness of our partitioning approach on dataset CHM1, VarMatch partitions 6,438,208 initial small variants into 3,272,206 subproblems, 99.9% of which have less than 9 variants in them. Figure 1b shows that on dataset CHM1 VarMatch scales with multiple threads, while with more threads I/O becomes the bottleneck($\sim$ 200 seconds).

References

1. Tan, A., Abecasis, G. R., and Kang, H. M. *Bioinformatics* , btv112 (2015).
2. Li, H. *Bioinformatics* **30**(20), 2841–2851 (2014).
3. Zook, J. M., Chapman, B., Wang, J., Mittelman, D., Hofmann, O., Hide, W., and Salit, M. *Nature biotechnology* **32**, 246–251 (2014).
4. Cleary, J. G., Braithwaite, R., Gaastra, K., Hilbush, B. S., Inglis, S., Irvine, S. A., Jackson, A., Littin, R., Rathod, M., Ware, D., et al. *bioRxiv* , 023754 (2015).