Scientific Reports, Vol. 10, 2020

SCIENTIFIC REPORTS
nature research

OPEN

Identifying multi-hit carcinogenic gene combinations: Scaling up a weighted set cover algorithm using compressed binary matrix representation on a GPU

Qais Al Hajri[1,5], Sajal Dash[2,5], Wu-chun Feng[1,2], Harold R. Garner[3,4] & Ramu Anandakrishnan[3,4,5*]

Despite decades of research, effective treatments for most cancers remain elusive. One reason is that different instances of cancer result from different combinations of multiple genetic mutations (hits). Therefore, treatments that may be effective in some cases are not effective in others. We previously developed an algorithm for identifying combinations of carcinogenic genes with mutations (multi-hit combinations), which could suggest a likely cause for individual instances of cancer. Most cancers are estimated to require three or more hits. However, the computational complexity of the algorithm scales exponentially with the number of hits, making it impractical for identifying combinations of more than two hits. To identify combinations of greater than two hits, we used a compressed binary matrix representation, and optimized the algorithm for parallel execution on an NVIDIA V100 graphics processing unit (GPU). With these enhancements, the optimized GPU implementation was on average an estimated 12,144 times faster than the original integer matrix based CPU implementation, for the 3-hit algorithm, allowing us to identify 3-hit combinations. The 3-hit combinations identified using a training set were able to differentiate between tumor and normal samples in a separate test set with 90% overall sensitivity and 93% overall specificity. We illustrate how the distribution of mutations in tumor and normal samples in the multi-hit gene combinations can suggest potential driver mutations for further investigation. With experimental validation, these combinations may provide insight into the etiology of cancer and a rational basis for targeted combination therapy.

Cancer is one of the leading causes of death in the US with a projected 606,880 deaths in 2019[1]. Despite significant progress, effective treatment in advanced cases remain elusive, with most progress coming from prevention and early detection[2,3]. One of many possible reasons is that, although cancer is known to be caused primarily by multiple genetic mutations[4–9], we cannot in general determine the specific combination of mutations responsible for a given instance of cancer[10,11]. Knowing the specific combination of hits in individual cases would allow us to develop more effective targeted combination therapies[10,11]. Although there are other factor that may contribute to cancer growth, such as tumor microenvironment, epigenetic modifications, gene fusion, germline defects, etc., the focus of this work is on somatic mutations[12–18].

Current computational approaches search for cancer genes and mutations that increase cancer risk (the probability of getting cancer)[18–29]. In addition to other considerations, these methods search for genes that are significantly more frequently mutated in tumor samples compared to an estimated background mutation rate. However, mutations in any one of these genes by themselves do not always result in cancer, suggesting that carcinogenesis may require additional mutations, as illustrated by the following exmples. Germline mutations in BRCA genes

[1]Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, 24060, USA. [2]Department of Computer Science, Virginia Tech, Blacksburg, VA, 24060, USA. [3]Department of Biomedical Sciences, Edward Via College of Osteopathic Medicine, Blacksburg, VA, 24060, USA. [4]Gibbs Cancer Center and Research Institute, Spartanburg, SC, 29303, USA. [5]These authors contributed equally: Qais Al Hajri, Sajal Dash and Ramu Anandakrishnan. *email: ramu@vt.edu

increase the risk of breast cancer. Although 72% of women with this mutation are likely to be diagnosed with breast cancer by age 80, the other 28% of women are unlikely to get the disease[4]. In addition, none of the women with this inherited mutation are likely to get cancer before age 20, suggesting that additional genetic defects (hits) may be required for carcinogenesis. Similarly, individuals with APC mutations have a 7% risk of developing familial adenomatous polyposis, representing colon cancer predisposition, by age 21, which increases to a 99% risk by age 80[5]. The Li Fraumeni syndrome is another example where germline P53 mutations are associated with early onset cancer predisposition (e.g. soft tissue and bone sarcomas). However, cancer penetrance is less than 20% for children while approaching 80% by age 70, indicating that multiple hits are required for carcinogenesis[6–8]. The classic example for the multi-hit hypothesis is the study of retinoblastoma by Knudson[9]. The study showed that mutations in a single copy of the RB1 gene increases the risk of retinoblastoma, but a second mutation in the other copy of the gene is required for carcinogenesis. Mathematical models based on cancer incidence and mutation data suggest that combinations of two–eight oncogenic mutations are required for carcinogenesis, depending on cancer type[30–37]. *In vitro* studies have investigated the role of combinations of multiple genetic defects[38–40]. However, these studies consider known combinations, and are not designed to discover novel combinations.

Unlike computational approaches that search for driver mutations and cancer genes that increase the risk of cancer[18–29], we previously developed an approach that explicitly searches for combinations of genes with mutations that are likely to be the cause of specific instances of cancer[41]. A set of 2-hit combinations identified by the algorithm was able to differentiate between tumor and normal tissue samples with 90% overall sensitivity and 93% overall specificity. Here, we extend the algorithm to identify combinations of three or more hits, since most cancers are estimated to require more than two hits[30–37]. However, the computational complexity of the algorithm, which is $O(G^h \times C \times (N_t + N_n))$, limits the combinations that can be practically identified to 2-hit ($h = 2$) combinations, where $G \approx 20000$ is the number of genes with mutations in the input data, C is the number of combinations identified by the algorithm, N_t is the number of input tumor samples, N_n is the number of input normal samples, and h is the number of hits. For example, it took 39 minutes to calculate a set of 2-hit combination for breast cancer (BRCA) using 911 tumor samples from the cancer genome atlas (TCGA). The algorithm was run on an Intel Xeon E5-2630 2.1 GHz central processing unit (CPU) with 256 GB memory. We estimate (as described in Methods) that it will take 253 days to calculate a set of 3-hit ($h = 3$) combinations for BRCA, without any additional optimization or parallelization.

The goal of this work is to optimize the multi-hit algorithm to identify combinations of more than two hits in a practical time frame (<1 month). Achieving this level of speedup requires parallel execution across a large number of processors. Graphical processing units (GPUs) with thousands of processors are a natural choice for massively parallel processing[42]. However, GPUs have three key limitations that must be addressed to achieve significant speedup. (1) Speed of memory access is significantly slower on GPUs compared to CPUs, e.g. on the Intel Xeon E5-2630 CPU L1 and L2 cache access require 4 and 11 cycles respectively[43], compared to 28 and 193 cycles for the NVIDIA V100 GPU[44]. Therefore, speedup from parallelization will be offset by slower memory access for algorithms that require access to a large amount of data from memory. (2) GPUs have limited amount of accessible memory, e.g. 32GB for the NVIDIA V100, compared to 1.5TB for Intel Xeon E5-2630[45]. (3) On NVIDIA GPUs, divergent branching during execution will result in unbalanced processor load, which also limits the achievable speedup from parallelization[46–50]. To address these GPU limitations, we employed two general strategies. (1) We used a compressed binary representation for the Gene-Sample Mutation matrix (described in Methods), which reduced memory requirement by 16-fold and resulted in an average 10 fold speedup (see Results). (2) We restructured and optimized the algorithm for parallel execution on a NVIDIA Tesla V100 PCIe graphical processing unit by minimizing divergent branching in addition to other optimizations described in the Methods section. The compressed binary representation alone resulted in a 0.4–18 fold speedup for the 2-hit algorithm, compared to the original integer matrix, depending on cancer type. This additional speedup, and the associated increase in software complexity, was not necessary for the identification of 2-hit combinations, and insufficient by itself for the identification of 3-hit combinations on the CPU. However, the optimized GPU implementation combined with the compressed binary representation was 0.7–224 times faster than the original CPU based integer matrix implementation, for the 2-hit algorithm, depending on cancer type. The 3-hit algorithm was an estimated 29–33,690 times faster for the optimized GPU implementation compared to the original CPU implementation. For the breast cancer samples mentioned above, we were able to compute a set of 3-hit combinations in 23 minutes with the optimized GPU implementation compared to the estimated 253 days for the original CPU implementation. The set of 3-hit combinations identified using a randomly partitioned training set was able to differentiate between tumor and normal samples in separate test data with overall sensitivity of 90% (95% confidence interval (CI) = 88–91%) and overall specificity of 93% (95% CI = 92–94%). Despite this relatively high accuracy, the multi-hit gene combinations identified by our algorithm may not represent cancer genes (see Discussion). Further experimental validation will be required to determine if mutations within these genes may play a role in cancer genesis or progression.

The remainder of this manuscript is organized as follows. In the Results section, we describe the speedup achieved by the optimized parallel implementation, the breakdown of the contribution of different optimizations, and the accuracy of the multi-hit combinations identified. In the Discussion section, we illustrate how the distribution of somatic mutations in tumor and normal samples in the gene combinations can be used to identify potential driver mutations for further investigation. Our approach and results are summarized in the Conclusions. In the Methods section, we describe the multi-hit algorithm, the compressed binary representation of the input matrix, the mapping of the algorithm to the GPU, and its optimization for parallel execution.

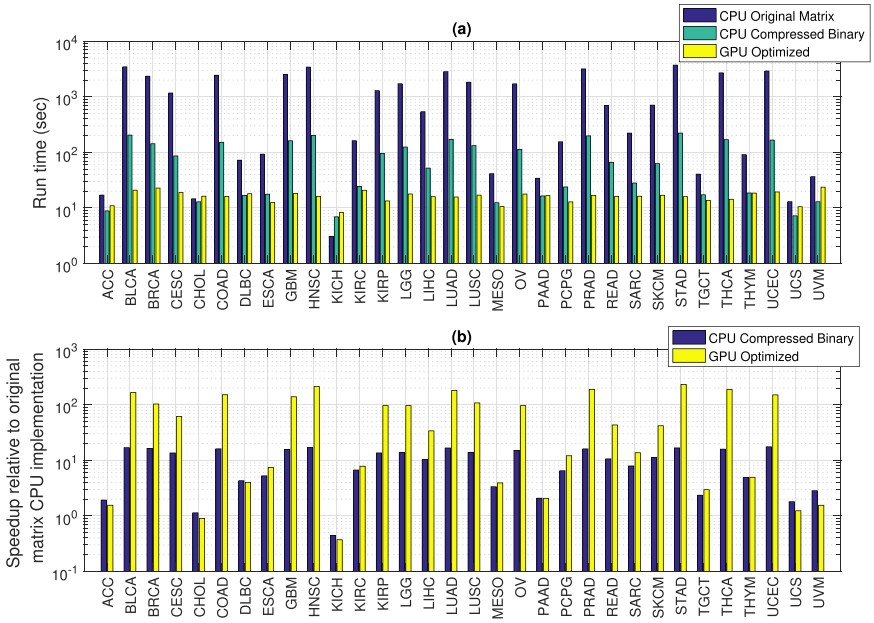

Figure 1. Comparison of different implementations of the multi-hit algorithm for identifying 2-hit combinations. (**a**) Run time for the original matrix implementation on the CPU ranges from 3–3723 sec compared to 7–223 sec for the compressed binary CPU implementation and 5–33 sec for the optimized GPU implementation. (**b**) Speedup is on average 10-fold for the compressed binary CPU implementation and 68-fold for the optimized GPU implementation compared to the original matrix CPU implementation. Names for the cancer types shown along the x-axis are listed in Table S1.

Results

Cancer is estimated to be caused by a combination of a small number of (two to eight) genetic mutations (hits)[30–37]. We had previously developed an algorithm for identifying a set of 2-hit combinations of genes with mutations, that was able to differentiate between tumor and normal samples with high sensitivity and specificity[41]. Due to its computational complexity the algorithm is impractical for identifying more than two hits[41].

To identify combinations of more than two hits, we restructured and optimized the algorithm for parallel execution on a GPU, as described in the Methods section. These modifications can be grouped into two broad categories: compressed binary matrix representation and GPU parallelization.

The compressed binary matrix optimization and GPU parallelization resulted in an average speedup of 12,144x for the 3-hit algorithm, relative to the original integer matrix based CPU implementation. With this speedup, we were able to identify 3-hit combinations for the 32 cancer types for which data was available in TCGA. In addition, we were able to identify 4-hit combinations for 14 cancer types for which the run time was less than 15 days. The accuracy of the 3-hit combinations was found to be comparable to the 2-hit combinations, with overall sensitivity of 90% (95% CI = 88–91%) and average specificity of 93% (95% CI = 92–94%).

Optimization and parallelization reduces run time for the 2-hit algorithm. Figure 1(a) shows that the run time for identifying 2-hit combinations ranges from 5–33 sec for the optimized GPU implementation compared to 7–223 sec for the compressed binary CPU implementation and 3–3,723 sec for the original matrix CPU implementation. The optimized GPU implementation of the 2-hit algorithm is on average 68 times faster than the original CPU implementation, with the speedup ranging from 0.7–224x (Fig. 1(b)). However, due to the relatively large fixed data load time, these speedup numbers understate the effect of the optimization and parallelization described in the Methods. On average, the data load time for the 2-hit optimized GPU implementation is 85% of the total run time. The speedup values for the 3-hit algorithm, where the above average data load times are 14% of total run time for the optimized GPU implementation, is more closely representative of the effect of optimization and parallelization. Detailed run times for each cancer type, with a breakdown of the data load time, for different implementations of the 2-hit algorithm are shown in Supplementary Table S2.

Run time reduction permits identification of 3-hit combination. Figure 2(a) shows that the run time for identifying 3-hit combinations ranges from 4 sec to 23 min for the optimized GPU implementation compared to 46 sec to 10 days for the compressed binary CPU implementation. For the original integer matrix CPU implementation, the run time ranges from 110 sec to an estimated 282 days. The optimized 3-hit algorithm on the GPU results in an estimated 29 –33,690 fold speedup compared to the estimated time for the original matrix based CPU implementation (Fig. 2(b)), with an average 12,144 fold estimated speedup. Detailed run times and speedup for each cancer type for different implementations of the 3-hit algorithm are shown in Supplementary Information Tables S4 and S5.

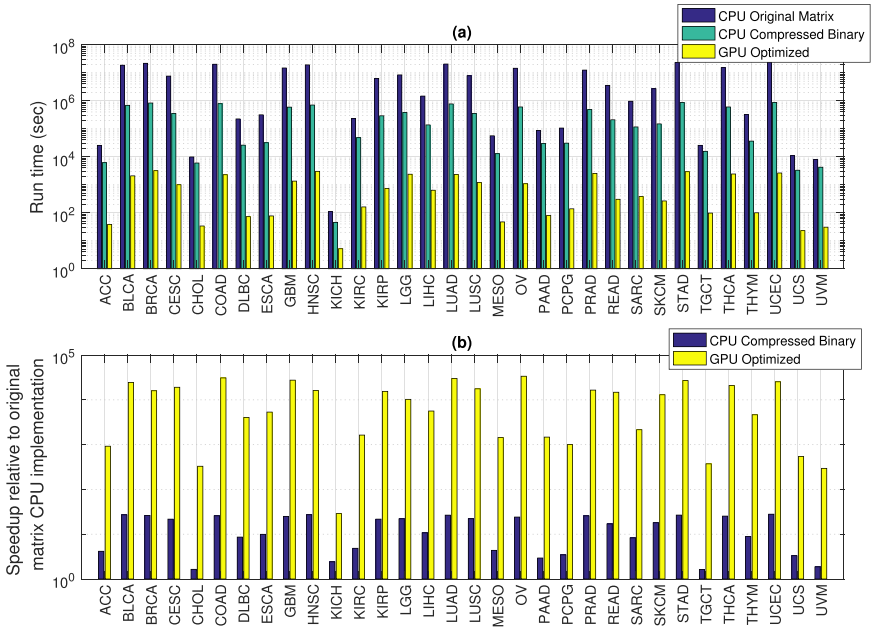

Figure 2. Comparison of different implementations of the multi-hit algorithm for identifying 3-hit combinations. (**a**) Run time for the original matrix CPU implementation ranges from 110 sec to an estimated 282 days, compared to 46 sec to 10 days for the compressed binary CPU implementation and 4 sec to 23 min for the optimized GPU implementation. Run times for the original matrix CPU implementation requiring over 30 days were estimated as described in Methods. (**b**) Speedup for the compressed binary CPU implementation ranged from 2x–28x, and from 29x–33,690x for the optimized GPU implementation. Names for the cancer types shown along the x-axis are listed in Table S1.

Run time reduction permits identification of some 4-hit combination. With the reduction in run time resulting from the optimization and parallelization described in the Methods section below, we were able to identify 4-hit combinations for some cancer types. For cancer types where the number of genes with mutations $G < 19000$, it takes less than 15 days to identify 4-hit combinations. Detailed run times for these cancer types are shown in Supplementary Information Table S6. To identify 4-hit combinations for all cancer types, additional optimization and parallelization across multiple GPUs will be required, which will be presented in a separate forthcoming study.

Contribution of optimization techniques to overall speedup. The speedup reported above results from five key enhancements: compressed binary representation of the Gene-Sample Mutation matrices, parallel execution across multiple GPU cores, removal of branch and bound logic, computation of a single two-gene combination per thread, and mapping upper triangular matrix of two-gene combinations to thread index. See Methods section below. The breakdown of the contribution due to each of these enhancements is shown in Fig. 3. The speedup contribution of each enhancement is calculated as the difference in average speedup for the implementation of each enhancement compared to the original matrix CPU implementation See Supplementary Tables S3 and S5. On average, the largest contribution to speedup for the 2-hit algorithm is due to GPU parallelization (Fig. 3(a)). The largest contribution for the 3-hit algorithm is due to mapping GPU threads to the upper triangular matrix of two-gene combinations (Fig. 3(b)). The contribution due to the first three factors – compressed binary representation, GPU parallelization and removal of branch and bound logic – is roughly consistent between the 2-hit and 3-hit algorithms. However, the enhancements for a single two-gene combination per thread and upper triangular thread mapping slow down the 2-hit algorithm. This is because, for the 2-hit algorithm, speedup due to higher processor utilization from these enhancements are offset by the additional operations and global memory access required to implement these modifications.

Multi-hit combinations differentiate between tumor and normal samples with high accuracy. The 3-hit combinations identified using a 75% randomly selected Training set identified an average of 7 combinations per cancer type with a total of 335 unique genes, compared to 8 combinations per cancer type with a total of 310 unique genes for the 2-hit combinations. The identified combinations are listed in Supplementary Tables S7–S9. The 3-hit combinations were able to differentiate between tumor and normal samples in a separate Test set with overall sensitivity of 90% (95% CI = 88–91%) and overall specificity of 93% (95% CI = 92–94%), as shown in Fig. 4(b). This was comparable to the overall sensitivity and specificity for 2-hit combinations with sensitivity = 90% (95% CI = 89–92%) and specificity = 94% (95% CI = 93–95%), as shown in Fig. 4(a). The difference in average sensitivity and specificity between 2- and 3-hit combinations was −6% (95% CI = −13.5–+1.5%) and −1% (95% CI = −3.6–+1.6%) respectively, with corresponding p-values of 0.12 and 0.44 respectively. Accuracy values are listed in Supplementary Tables S10 and S11. Since we did not see any improvement in accuracy for 3-hit

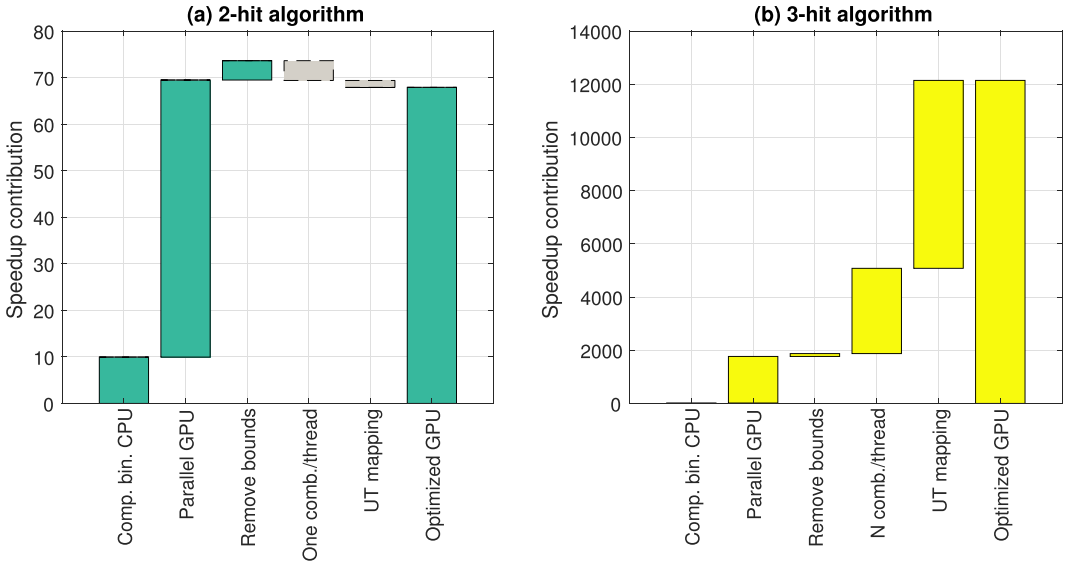

Figure 3. Average contribution of optimizations and parallelization to speedup. Breakdown of contributions due to compressed binary representation, GPU parallelization, removal of branch and bound logic, single two-gene combination per thread, and mapping of upper triangular (UT) gene combination to a sequential thread ID. (**a**) Breakdown of 2-hit speedup. (**b**) Breakdown of 3-hit combinations. Contribution due to compressed binary representation is 15x for 3-hits which is not visible in the scale of the figure.

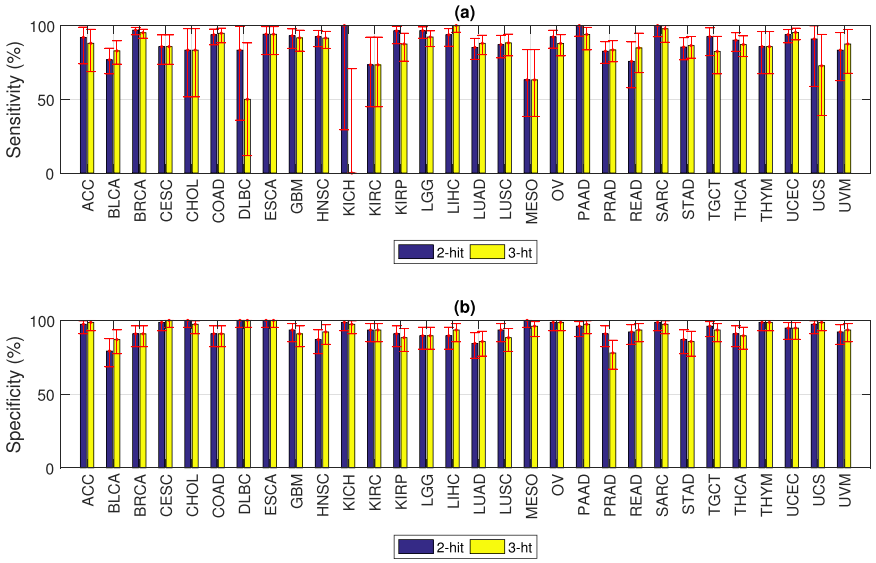

Figure 4. Accuracy of 2- and 3-hit combinations. (**a**) Sensitivity varies from 63–100% for 2-hit combinations, and from 50–100% for 3-hit combinations, excluding KICH for which there were only a total of 9 tumor samples. (**b**) Specificity varies from 79–100% for 2-hit combinations, and from 78–100% for 3-hit combinations. Sensitivity and specificity were calculated on a randomly selected 25% Test data set. Error bars represent 95% CI. Cancer types with relatively large 95% CI (CHOL, DLBC, KICH, KIRP, MESO and UCS) are due to small sample size (total of 44, 43, 9, 88, 69 and 46 samples respectively).

combinations compared to 2-hit combinations, we speculate that additional accuracy improvement will require examining individual mutations within genes, as discussed below.

Discussion

Not all mutations within a cancer gene are oncogenic[22,23,25,28]. However, to make the problem of identifying multi-hit combinations tractable, the algorithm searched through all possible gene combinations, instead of all possible combinations of mutations. In the tumor sample data used, there were over 400,000 unique somatic mutations across ~20,000 genes. It is theoretically possible to search all possible combinations of 400,000 protein altering somatic mutations instead of combinations of 20,000 genes with somatic mutations. However, searching

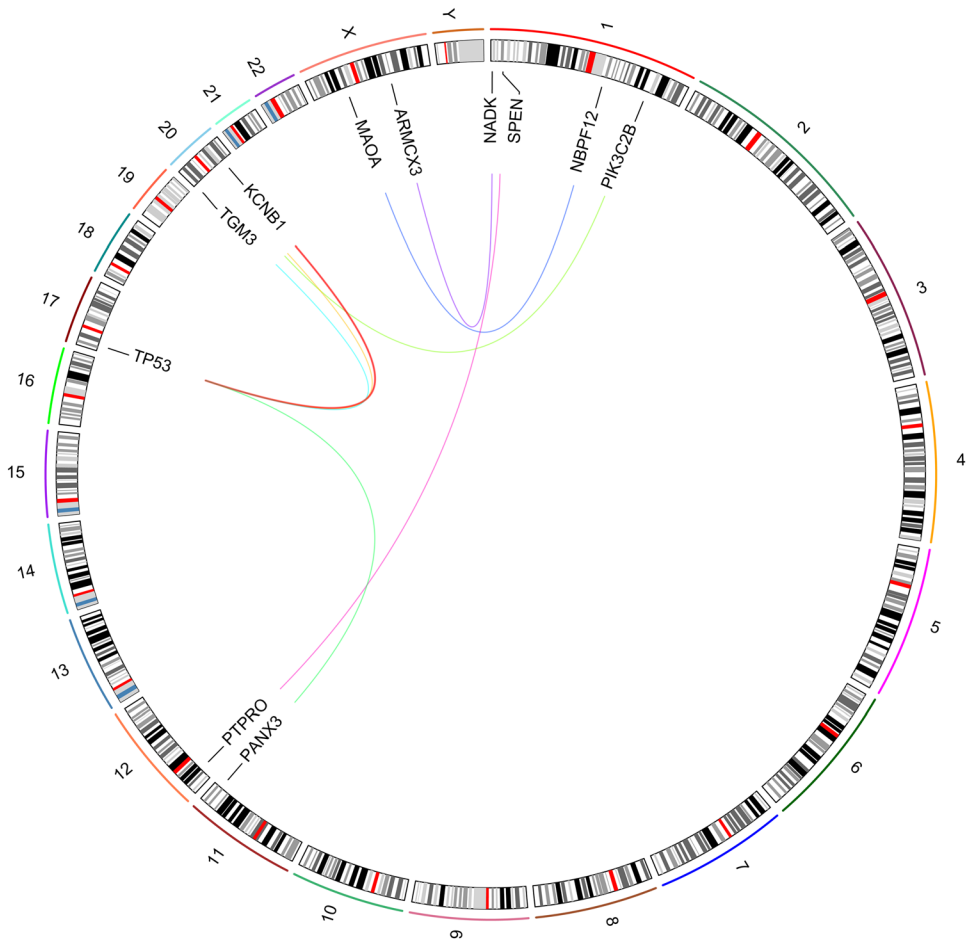

Figure 5. 2-hit combinations identified for ovarian serous cystadenocarcinoma (OV). The outer circle shows individual chromosomes with corresponding ideograms shown in the inner circle. Genes that comprise 2-hit combinations are labeled inside the circle. Each 2-hit combination is identified by differently colored lines connecting two genes. The red line represents the gene combination discussed in further detail. This image was generated using RCircos[87].

all possible combinations of 400,000 mutations would increase the computational complexity for identifying 3-hit combinations by over six orders of magnitude, making the problem computationally intractable. In addition, since there can be multiple different carcinogenic mutations within a gene, combinations of individual mutations will occur less frequently than combinations of genes with mutations, further increasing the challenge of identifying carcinogenic combinations within this much larger set of possible combinations. Therefore, we chose to first focus on combinations of genes with mutations. Mutations within these gene combinations can then be examined to identify potential driver mutations for further investigation, as illustrated below.

Consider for example, the 2- and 3-hit combinations identified for ovarian serous cystadenocarcinoma (OV) (Figs. 5 and 6). The most commonly occurring 2- and 3-hit combination are TP53+KCNB1 and TP53+KCNB1+TTN respectively. Mutations in TP53 and KCNB1 occur in 279 of 317 OV tumor samples and mutations in TP53, KCNB1 and TTN occur in 271 of 317 OV tumor samples. The distribution of protein altering somatic mutations in TP53, KCNB1 and TTN for the 271 OV tumor samples containing mutations in all three genes are shown in Figs. 7(a), 8(a) and 9(a), respectively. The distribution of protein altering somatic mutations in TP53, KCNB1 and TTN for 333 normal samples are shown in Figs. 7(b), 8(b) and 9(b), respectively. The difference in the frequency of individual mutations between tumor and normal samples may suggest potential driver mutations for further investigation.

The TP53 gene codes for the Tumor Protein P53. Mutations in TP53, a tumor suppressor gene, have been extensively implicated in many cancers, including OV[51–56]. In the 271 OV tumor samples containing the TP53+KCNB1+TTN 3-hit combination, TP53 contains on average 1.8 protein altering somatic mutations per sample, compared to 0.15 mutations per sample in normal samples, with clear differences in the distribution of these mutations (Fig. 7). The three most frequently occurring mutations in the tumor samples (amino acid positions R248, R273, and R175) are potential driver mutations, since they rarely occur in normal tissue (Fig. 7). The mutation frequency at R248, R273 and R175 are 0.08, 0.07 and 0.06 per tumor sample, compared to 0.00 per normal sample (p-value < 0.0001 for the difference in proportions). In fact, previous studies have shown that the R248W, R273H and R175H mutations not only cause a loss of P53-based tumor suppressor activity, but also result

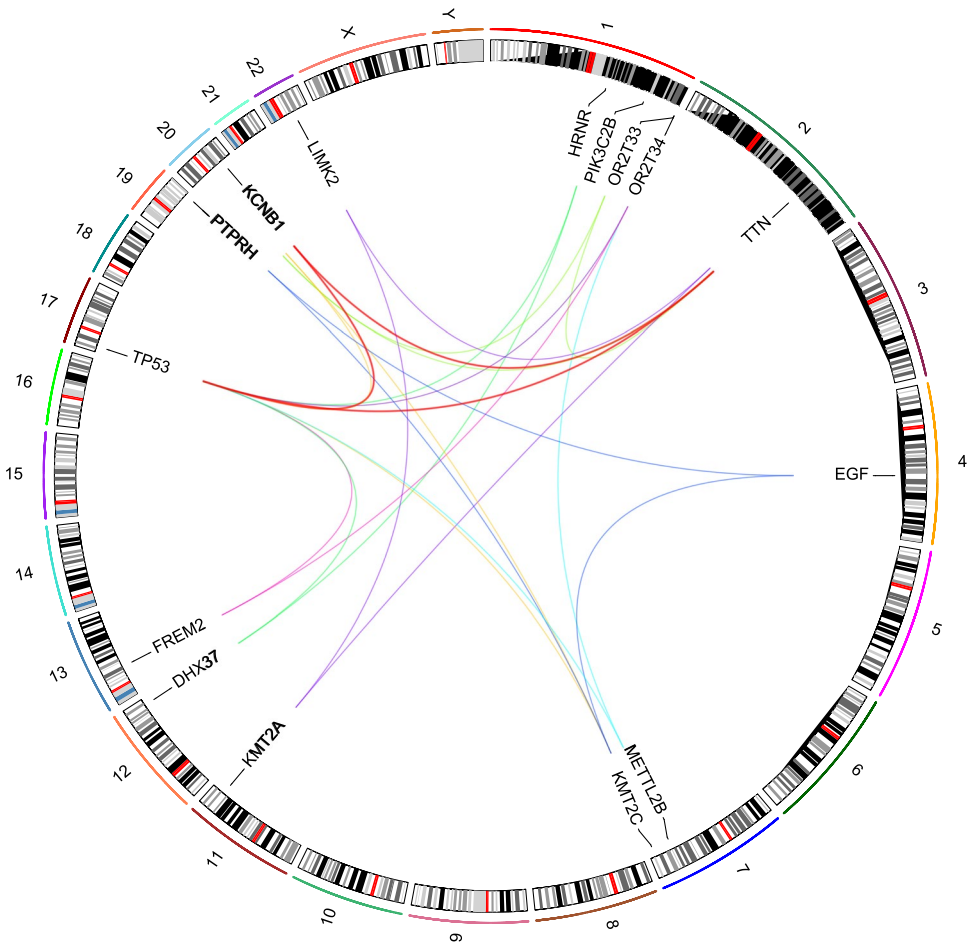

Figure 6. 3-hit combinations identified for ovarian serous cystadenocarcinoma (OV). The outer circle shows individual chromosomes with corresponding ideograms shown in the inner circle. Genes that comprise 3-hit combinations are labeled inside the circle. Each 3-hit combination is identified by differently colored lines connecting three genes. The red line represents the gene combination discussed in further detail. This image was generated using RCircos[87].

in genomic instability causing gain of oncogenic activity[57–59]. On the other hand the two most frequently mutated amino acid positions in normal samples, T377 and T378, are likely to be passenger mutations. Normal tissue mutation frequencies of 0.07 and 0.05 per normal sample are comparable to tumor tissue mutation frequencies of 0.04 and 0.05 per tumor sample for T377 and T378, respectively (Fig. 7).

The KBNB1 gene codes for the Potassium Voltage-Gated Channel Subfamily B Member 1 protein. KCNB1 has been previously identified as a prognostic factor in gliomas due to its tumor suppressor function[60]. It contains on average 2.14 protein altering somatic mutations per tumor sample in the 271 OV samples containing the TP53+KCNB1+TTN 3-hit combination, compared to 0.03 mutations per normal sample (p-value < 0.0001) (Fig. 8). The two most frequently occurring mutations at K776 and R736 are potential driver mutations worthy of further investigation. The mutation frequencies at these positions are 1.37 and 0.41 per tumor sample compared to 0.00 and 0.003 per normal sample, respectively (Fig. 8). Although KCNB1 has been extensively studied, primarily in the context of epilepsy[61–66], these studies do not include either of the two mutations identified here. These two mutations occur in the unstructured C-terminus cytoplasmic tail region of this transmembrane potassium channel protein[61,66]. Further *in vitro* investigation will be required to understand how these mutations may affect the expression, structure or function of this protein, to determine if these could be driver mutations.

The TTN gene codes for the Titin protein of striated muscle. TTN expression level has been previously identified as prognositc marker for Ewing's sarcoma[67], and TTN mutations have been associated with several myopathies[68–72]. Titin is a large protein consisting of 34,350 amino acids, with a correspondingly large number of mutations, 15.37 protein altering somatic mutations per tumor sample and 3.98 mutations per normal sample, on average (Fig. 9). Three of the most frequent mutations in TTN in tumor samples, C21862G, E1656G and T2963P, occur more frequently in tumor samples compared to normal samples, suggesting that these may be potential driver mutations that should be investigated further. The mutation frequencies at these amino acid positions are 0.17, 0.20 and 0.20 per tumor sample, compared to 0.06, 0.003, and 0.03 per normal tissue sample, respectively (Fig. 9). Although TTN mutations have been extensively studied, primarily in the context of myopathies[69–72], these studies do not include any of the three mutations identified here.

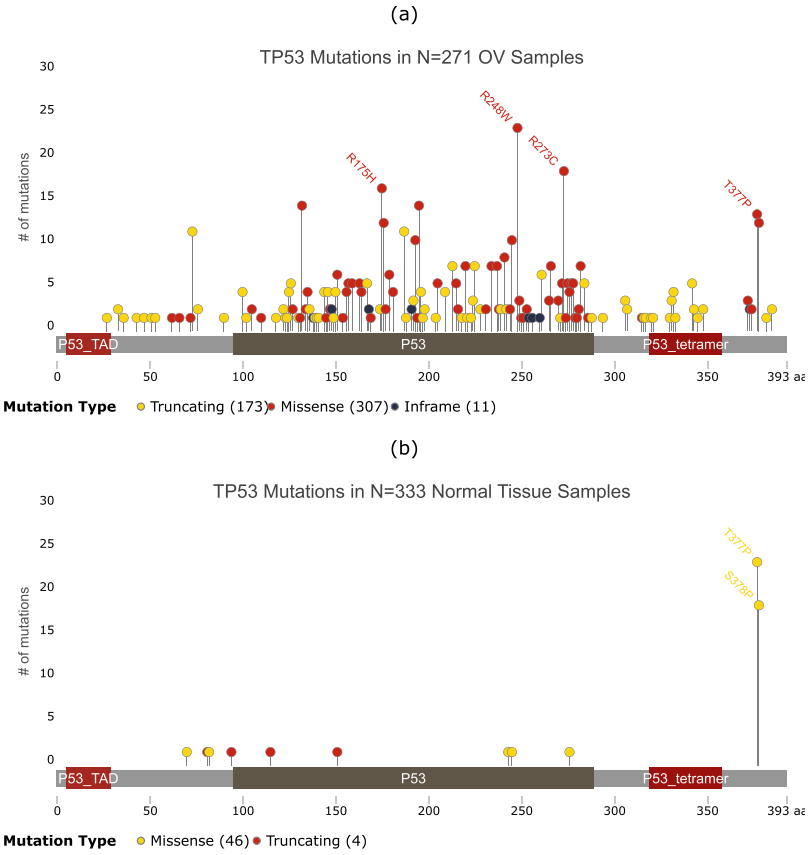

Figure 7. Distribution of somatic mutations in TP53 in ovarian tumor samples and normal samples. The horizontal bar shows amino acid position within the protein, with labels showing known functional domains. Vertical lines show the number of samples with protein altering mutations at each amino acid position. The most frequently mutated sites for each gene in (**a**) tumor and (**b**) normal samples are labeled for comparison. Image generated using g3viz[88].

The above only provide a starting point for further investigation. The positive selection implied by the higher mutation frequencies seen above are confounded by several factors, including tumor microenvironment, tissue and cell type, epigenetic modifications, gene expression and co-expression, etc.[12–18]. A more detailed analysis of the potential driver mutations identified above using available literature, gene expression data, copy number variation, associated pathways, functional annotation, protein localization, etc., could provide additional evidence to either support or reject the mutation as a driver mutation. This information can be iteratively incorporated into the search algorithm described in Methods. We expect that excluding likely passenger mutations will reduce the number of false positives and prioritizing likely driver mutations will reduce false negatives, improving the accuracy of the combinations identified. However, this could also potentially limit the discovery of novel genes.

Note that these somatic mutations were calculated using *protected* whole exome sequencing data from tumor samples with matched blood-derived normal samples. For tumor samples, protected somatic mutation data (MAF files) were downloaded from the cancer genome atlas (TCGA) with permission. Somatic mutations for normal tissue samples with matched blood-derived normal samples were called using the same protocol used by TCGA, as described in the methods. Variants called using matched blood-derived normal data identifies significantly more mutations than the number of variants called without matched blood-derived normal samples, for the following reasons[73,74]. Biopsy specimens contain a mix of tumor and normal tissue cells, tumor-infiltrating lymphocytes, and stromal cells. In addition, tumor cells themselves can be genetically diverse. As a result, mutations in a subset of the cell population will present at a relatively low frequency. Using blood derived normal samples as a reference allows for the identification of such low frequency variants. Variants that could potentially lead to de-identification of donors (~80 million variants) are considered "protected" data in TCGA, and are not accessible by tools such as cBioPortal and TCGA queries that are based on "open" access data (~3 million variants). For example, the protected TCGA MAF files contain 617 protein-altering somatic mutations in TP53 in 317 OV samples, compared to the 276 somatic mutations reported by cBioPortal using open access data[75].

Conclusion

Cancer is caused by a combination of a small number of genetic defects (hits), estimated to be in the range of two to eight. However, the specific multi-hit combination for each instance of cancer can be different, even for the same type of cancer. Existing approaches focused on identifying individual cancer genes can not identify the specific multi-hit combination responsible for an individual instance of cancer. We previously developed a

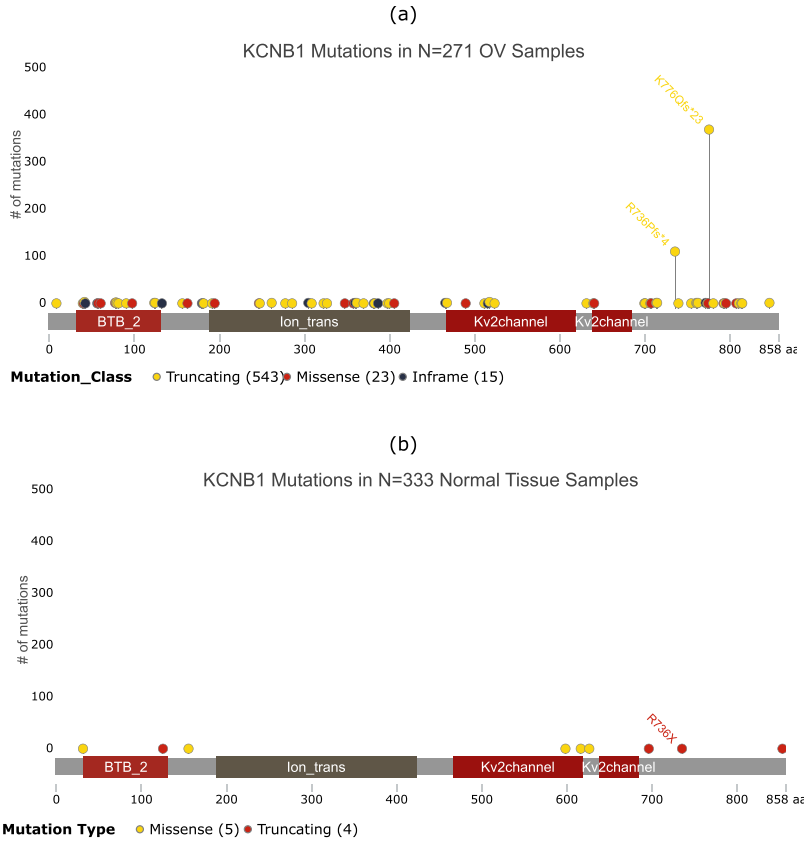

Figure 8. Distribution of somatic mutations in KCNB1 in ovarian tumor samples and normal samples. The horizontal bar shows amino acid position within the protein, with labels showing known functional domains. Vertical lines show the number of samples with protein altering mutations at each amino acid position. The most frequently mutated sites for each gene in (**a**) tumor and (**b**) normal samples are labeled for comparison. This image was generated using g3viz[88].

fundamentally different approach focused on identifying multi-hit combinations. Due to the $O(G^h)$ scaling of the search algorithm, where $G \approx 20000$ is the number genes and h is the number of hits, we were limited to identifying 2-hit combinations. In this work we present optimization and parallelization techniques that allowed us to extend the algorithm to identify 3-hit combinations, and some 4-hit combinations. The 3-hit combinations are able to differentiate between tumor and normal samples with overall 90% sensitivity (95% CI = 88–91%) and 93% specificity (95% CI = 92–94%). We illustrate how the distribution of somatic mutations in these genes can be used to identify potential driver mutations for further investigation. For example, we identified two protein-altering somatic mutations in the KCNB1 gene which occur significantly more frequently in TCGA ovarian cancer samples compared to normal samples (p-value < 0.0001), suggesting that these mutations may be positively selected for in ovarian cancer. However, further experimental validation is required to determine if these mutations represent novel cancer driver mutations, or are simply passenger mutations. The muti-hit combinations identified here, with experimental validation, can be used to identify the specific cause of individual instances of cancer, allowing for the rational design of more effective targeted combination therapies.

Methods

The multi-hit algorithm identifies combinations of genes with mutations that may represent the potential cause for individual instances of cancer. Due to its computational complexity, the algorithm was limited to identifying combinations of two hits. To identify combinations of more than two hits, the algorithm was restructured and optimized for parallel execution as described below.

The multi-hit algorithm. The problem of identifying a set of multi-hit combinations of genes with mutations that are most likely to be responsible for individual instances of cancer can be mapped to a weighted set cover (WSC) problem[41]. The WSC problem falls into a class of "NP-complete" problems for which there is no known polynomial-time solution, but given a solution it can be verified in polynomial time[76,77]. In addition, problems in this class can be mapped to each other in polynomial time, such that if a polynomial time solution is found for any one of these problems, all of these problems can be solved in polynomial time[78,79]. Although the WSC problem is computationally intractable for our problem size, a near-optimal approximate solution can be found using a greedy algorithm. A greedy algorithm for the WSC problem was adapted for the problem

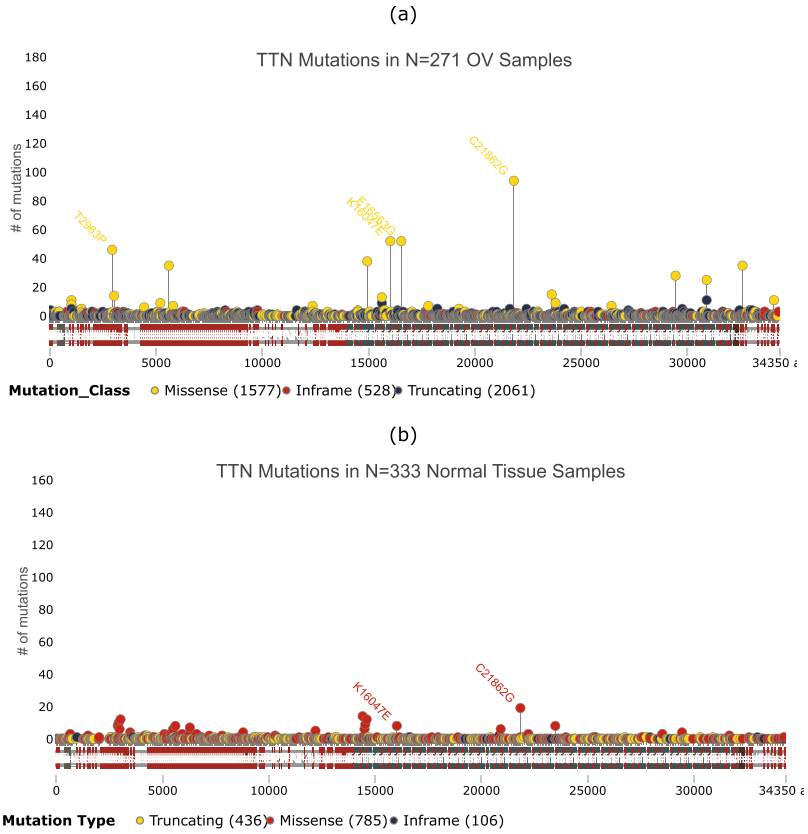

Figure 9. Distribution of somatic mutations in TTN in ovarian tumor samples and normal samples. The horizontal bar shows amino acid position within the protein, with labels showing known functional domains. Vertical lines show the number of samples with protein altering mutations at each amino acid position. The most frequently mutated sites for each gene in (**a**) tumor and (**b**) normal samples are labeled for comparison. This image was generated using g3viz[88].

of identifying a set of multi-hit combinations as previously described[41]. To summarize, the algorithm iterates through the following three steps until all tumor samples have been excluded, as illustrated in Fig. 10.

1. Compute a weighted accuracy metric F_i for all $i = [1, H]$ possible h-hit combinations, where H is the number of possible combinations. F_i is a combined measure of the specificity and sensitivity with which each combination can differentiate between tumor and normal samples in a training set.

$$F_i = \frac{\alpha TP_i + TN_i}{N_t + N_n} \tag{1}$$

   where, for a given combination i, TP_i is the number of true positives (tumor samples with mutations in the gene combination i), TN_i is the number of true negatives (normal samples without mutations in the gene combination i), N_t is the total number of tumor samples, N_n is the total number of normal samples and $\alpha = 0.1$ is a weighting factor to balance the contribution of sensitivity and specificity to the metric.
2. Select the combination of hits with the maximum F_i value, and add it to the list of selected multi-hit combinations.
3. Exclude all tumor samples that contain mutations in this combination of genes, from further consideration.

The computational complexity of the algorithm is $O(G^h \times C \times (N_t + N_n))$ where G is the number of genes and C is the number of combinations selected. The input to the algorithm are two Gene-Sample Mutation matrices, a tumor mutation matrix $(M^t_{ij}) \in \{0, 1\}^{G \times N_t}$ and a normal mutation matrix $(M^n_{ij}) \in \{0, 1\}^{G \times N_n}$. Non zero values in these binary matrices represent mutations in gene g_i, $i = [1, G]$ within sample s_j, $j = [1, N_t]$ for tumor samples and $j = [1, N_n]$ for normal samples (Fig. 10). In addition, these are sparse matrices with only 2% of the elements having a non-zero values. To take advantage of these characteristics of the input matrices, we considered two possible alternatives to the matrix representation: indexed array and compressed binary representations, as described below.

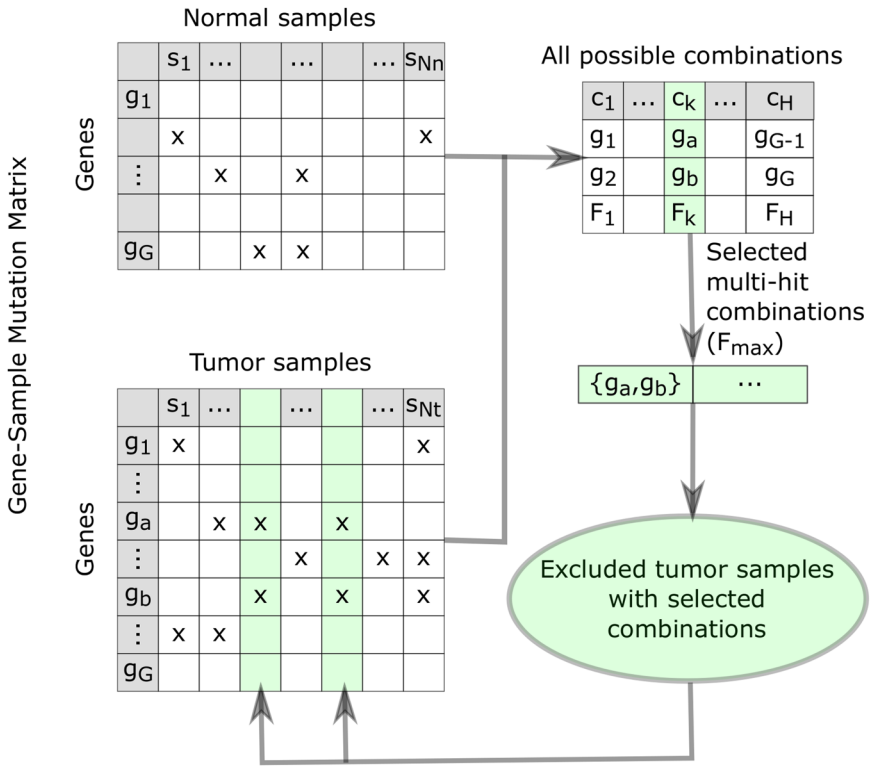

Figure 10. Algorithm for finding multi-hit combinations, illustrated for 2-hit combinations. The cells marked with x in the Gene-Sample Mutation matrices represent samples with mutations in the corresponding gene. There are $H = G(G-1)/2$ possible 2-hit combinations involving two different genes. The algorithm iterates through three steps. (1) Eq. (1) is used to calculate F_i for each combination. (2) The combination (g_a and g_b in this example) with the maximum value of F_i, (F_k in this example) is added to the list of selected multi-hit combinations. (3) Tumor samples containing mutation in the selected combination of genes are excluded from consideration in subsequent iterations of the algorithm. The green shaded columns in the Tumor Gene-Sample Mutation matrix represent excluded samples in the iteration shown. The algorithm terminates when all tumor samples have been excluded, i.e. "covered" by the set of multi-hit combinations.

Indexed array data structure. One option for speeding up the above algorithm's runtime, is to replace the Gene-Sample Mutation matrices with an indexed array data structure. The indexed array data structure reduces the number of arithmetic operations by excluding samples in the gene that do not have mutations. The data structure consists of two arrays. One is a samples array where samples with mutations within each gene are listed sequentially. The second is a gene index array, which contains the starting index into the samples array for each gene. With the indexed array representation, the algorithm would only examine samples with mutations in the genes being considered, instead of all samples. It is more efficient than the original matrix representation since samples that do not contain mutations in a gene are not evaluated. On average only 2% of samples have mutations for a given gene, therefore, we expected a significant speedup with an indexed list representation However, due to an increase in the number of instructions and divergent branches, the speedup using this data structure was less than what was achieved using the compressed binary representation described below.

Compressed binary representation. The binary values in the Gene-Sample Mutation matrices permits a reduction in memory requirement using a compressed binary representation. In addition, bitwise operations can be used with the compressed binary representation to reduce computational cost and divergent branching (discussed in section 5.5.2). Figure 11 illustrates how mutations in a group of four samples can be compressed into four bits. In the original implementation, each Gene-Sample value was represented as a single 16-bit short integer[41]. For this implementation, we represent groups of 64 samples as a single 64-bit unsigned integer, which requires 4-fold fewer vector operations compared to the 16-bit unsigned integer representation. The resulting speedup was confirmed experimentally (results not shown). The compressed binary representation also results in 16-fold reduction in memory since each word of memory stores data for 16 samples, compared to one sample per word in the original integer matrix representation.

The number of samples with mutations in a combination of genes can then be efficiently determined by a bitwise *AND* operation followed by a count of the non-zero bits, as illustrated in Fig. 11. To count the number of non-zero bits for the CPU code, we implemented Brian Kernighan's algorithm[80]. For the GPU implementation, we used the built-in `popcll()` function to count the number of bits set to 1 in the 64 bit unsigned integer. This function was faster than our own implementation of Brian Kernighan's algorithm (e.g. run time for optimized

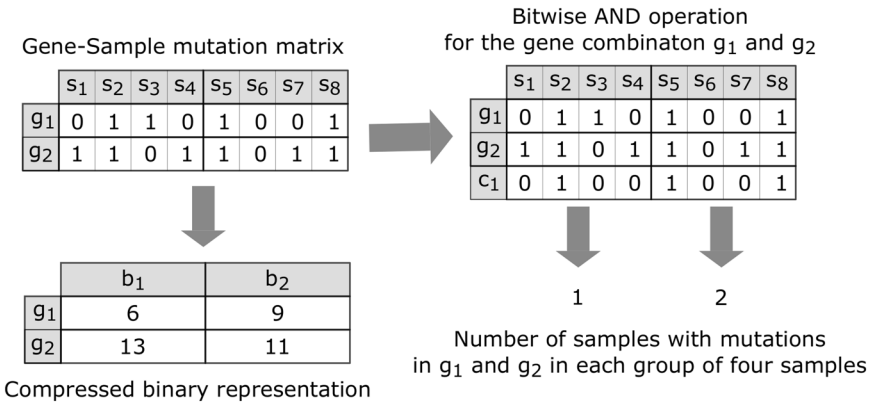

Figure 11. Compressed binary representation and bitwise operation for determining the number of samples with mutations in a combination of two genes. *Left:* Compressed binary representation of Gene-Sample Mutation matrices, illustrated for a 4-bit unsigned integer. s_i represents the normal or tumor samples shown in Fig. 10. Elements with 0 in the matrix indicate that the sample does not contain mutations in the corresponding gene, while 1 indicates that the sample does contain a mutation in the corresponding gene. Mutations in four samples can be represented by a single 4-bit unsigned integer. *Right:* Given any two genes g_i, g_j, the number of samples containing mutations in both these genes is determined by a bitwise `AND` operation for each of the integers representing mutations in g_i with the corresponding set of integers for g_j, and then counting the number of non-zero bits.

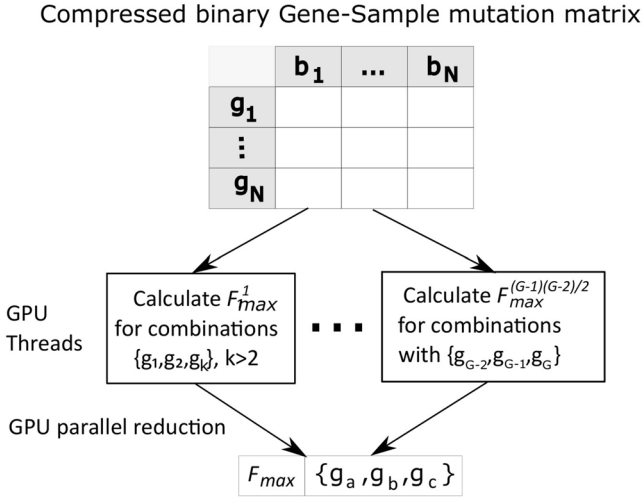

Figure 12. Mapping the multi-hit CPU algorithm to the GPU, illustrated for the 3-hit algorithm with the compressed binary representation (Fig. 11). Each GPU thread computes F^i_{max} for a subset of all possible combinations. The results of each thread is stored in GPU global memory. F_{max} across all subsets of combinations is calculated using parallel reduction[81].

GPU implementation for the 3-hit algorithm for BRCA using `popcll()`, is 451 sec faster than the run time using Kernighan's algorithm).

For the computation of 2-hit combinations for breast cancer (BRCA), the compressed binary representation resulted in a 16-fold speedup on the CPU compared to the original matrix representation, as shown in the Results. Since this speedup was considerably larger than the corresponding 4-fold speedup for the indexed array representation described above, we did not consider the indexed array structure any further.

Mapping to the NVIDIA Tesla V100 PCIe graphical processing unit (GPU). To further speed up the multi-hit algorithm, we restructured the CPU code for parallel execution on one GPU, specifically the NVIDIA Tesla V100 PCIe GPU[42]. The V100 consists of 5376 32-bit floating point cores, 5376 32-bit integer cores, 2688 64-bit floating point cores, 672 tensor cores and 336 texture units. The cores are partitioned into 84 streaming multiprocessors (SM) with 128 KB of shared memory per SM, 6 MB of *L2* cache and 32 GB of global memory for the GPU. Each SM is further partitioned into four single instruction multiple thread (SIMT) *warps*, i.e. the same instruction is executed on all cores within a warp, with each core running a different thread[42]. For parallel execution of the algorithm, we partition the computation of F_{max} across multiple threads, where each thread computes

the maximum value of F for a subset of combinations (F^i_{max}) where $i \in [1,\ G(G-1)/2]$. For 2-hit combinations, each thread processes a single combination, therefore $F^i_{max} = F^i$ for the combination i, say g_a and g_b where $a < b \leq G$. For 3-hit combinations F^i_{max} is the maximum value for all 3-hit combinations with two of the hits corresponding to 2-hit combination i, i.e. g_a, g_b and g_c where $a < b < c < G$, as illustrated in Fig. 12. The maximum value F_{max} across all F^i_{max} is then calculated using parallel reduction[81].

The sequential implementation of the above algorithm for 3-hit combinations is illustrated in Algorithm 1. The `for` loops in lines 7, 8, and 9, in the sequential algorithm iterates through all possible $\binom{G}{3}$ 3-hit combinations. Lines 10–14 compute F for one such combination. Lines 16–18 compute overall best combination along with it's F_{max} value. The remaining part of the algorithm updates excluded samples.

To run on parallel compute units of a GPU, we modified the above sequential algorithm as illustrated in Algorithm 2. We combined the two outer `for` loops into a single one, which iterates $\binom{G}{2}$ times (Line 7). Each iteration of this combined `for` loop can run in parallel on a different GPU compute units. For each of these parallel tasks, indexed by λ, there is a sequential `for` loop (Line 11) which computes the best combination among the 3-hit combinations that start with i, j corresponding to the λ (2). The mapping of λ to i and j in lines 7 and 8 is described below under "Minimizing divergent branches". At the end of this outer `for` loop, our parallel algorithm performs a parallel reduction (Line 25) to compute the best combination[81]. Then the tumor samples covered by this best combination are added to the covered samples.

This parallelization allows us to run $\binom{G}{2}$ parallel tasks with load $O(G(N_t + N_n))$ instead of $\binom{G}{3}$ sequential tasks with load $O(N_t + N_n)$.

Algorithm 1. Sequential algorithm to compute 3-hit combinations.

Require: *tumor-sets*, *normal-sets*, *tumor-samples*, *normal-samples*, α

1: $covered\text{-}samples \leftarrow \Phi$
2: $combinations \leftarrow \Phi$
3: $N_t \leftarrow |tumor\text{-}samples|$
4: $N_n \leftarrow |normal\text{-}samples|$
5: **while** $covered\text{-}samples \neq tumor\text{-}samples$ **do**
6: $\quad F_{max} \leftarrow 0.0$
7: $\quad$ **for** $i = 1 \rightarrow G$ **do**
8: $\quad\quad$ **for** $j = i+1 \rightarrow G$ **do**
9: $\quad\quad\quad$ **for** $k = j+1 \rightarrow G$ **do**
10: $\quad\quad\quad\quad TP \leftarrow |tumor\text{-}sets[i]) \cap tumor\text{-}sets[j]) \cap tumor\text{-}sets[k])|$
11: $\quad\quad\quad\quad FP \leftarrow |normal\text{-}sets[i]) \cap normal\text{-}sets[j]) \cap normal\text{-}sets[k])|$
12: $\quad\quad\quad\quad TN \leftarrow N_n - FP$
13: $\quad\quad\quad\quad FN \leftarrow N_t - TP$
14: $\quad\quad\quad\quad F \leftarrow \left(\frac{\alpha \times TP + TN}{N_t + N_n}\right)$
15:
16: $\quad\quad\quad\quad$ **if** $F \geq F_{max}$ **then**
17: $\quad\quad\quad\quad\quad F_{max} \leftarrow F$
18: $\quad\quad\quad\quad\quad best\text{-}combination \leftarrow \langle i, j, k \rangle$
19: $\quad\quad\quad\quad$ **end if**
20: $\quad\quad\quad$ **end for**
21: $\quad\quad$ **end for**
22: $\quad$ **end for**
23:
24: $\quad combinations.add(best\text{-}combination)$
25: $\quad i, j, k \leftarrow best\text{-}combination.extract()$
26: $\quad covered\text{-}samples.add(tumor\text{-}sets[i]) \cap tumor\text{-}sets[j] \cap tumor\text{-}sets[k])$
27: **end while**
28: **return** *combinations*

GPU optimization. The differences between the CPU and GPU architectures make certain coding techniques, that may be appropriate for serial execution on a CPU, sub-optimal or even incorrect for parallel execution on a GPU. Three critical considerations for minimizing processor latency are: synchronizing update access to memory locations shared by multiple processors, divergent branching in a single instruction multiple thread (SIMT) architecture, and relative speed of shared memory vs. global memory.

Minimizing GPU synchronization. A straightforward implementation would have a single global memory location for F_{max} which could be updated by all thread. However, such an implementation would require a synchronization or locking protocol to maintain cache coherence[82]. Synchronization of GPU threads to ensure correct results introduced significant processor latency and resulted in the GPU implementation running slower than the CPU version. We therefore allocate a separate memory location for each thread i to store its F^i_{max} value (Fig. 12), avoiding the need for synchronized memory access. We then use parallel reduction to efficiently calculate the global F_{max} value[81].

Algorithm 2. Parallel algorithm to compute 3-hit combinations.

Require: *tumor-sets*, *normal-sets*, *tumor-samples*, *normal-samples*, α

covered-samples $\leftarrow \Phi$
combinations $\leftarrow \Phi$
$N_t \leftarrow |tumor\text{-}samples|$
$N_n \leftarrow |normal\text{-}samples|$
while *covered-samples* $\neq$ *tumor-samples* **do**
  $F_{max}[1 \ldots \binom{G}{2}] \leftarrow [-\infty \cdots -\infty]$
  for parallel: $\lambda = 1 \rightarrow \binom{G}{2}$ **do**
    $F_{max}[\lambda] \leftarrow -\infty$
    $j \leftarrow \left\lfloor \sqrt{1/4 + 2\lambda} + 1/2 \right\rfloor$
    $i \leftarrow \lambda - j(j-1)/2$
    for $k = j+1 \rightarrow G$ **do**
      $TP \leftarrow |tumor\text{-}sets[i]) \cap tumor\text{-}sets[j]) \cap tumor\text{-}sets[k])|$
      $FP \leftarrow |normal\text{-}sets[i]) \cap normal\text{-}sets[j]) \cap normal\text{-}sets[k])|$
      $TN \leftarrow N_n - FP$
      $FN \leftarrow N_t - TP$
      $F \leftarrow \left(\frac{\alpha \times TP + TN}{N_t + N_n} \right)$

      if $F \geq F_{max}[\lambda]$ **then**
        $F_{max}[\lambda] \leftarrow F$
        $best\text{-}combinations[\lambda] \leftarrow \langle i, j, k \rangle$
      end if
    end for
  end for

  $best\text{-}combination, F_{max} \leftarrow parallel\text{-}reduction(best\text{-}combinations[1 \ldots \binom{G}{2}], F_{max}[1 \ldots \binom{G}{2}])$
  $combinations.add(best\text{-}combination)$
  $i, j, k \leftarrow best\text{-}combination.extract()$
  $covered\text{-}samples.add(tumor\text{-}sets[i]) \cap tumor\text{-}sets[j] \cap tumor\text{-}sets[k])$
end while
return *combinations*

Minimizing divergent branches. In the SIMT warps used by the GPU within streaming multiprocessors (SM), divergent branches introduce significant processor latency[46–50]. Divergent branches are `IF-ELSE` and `LOOP` control statements that cause execution along different paths depending on conditional values. Within a warp, all possible execution paths are serialized and evaluated[42,49,50]. Thus, significant latency is introduced due to the execution of instruction within branches that are not used. In the original integer matrix CPU implementation, conditional statements are required to count the number of samples with mutations in a gene combination. In the compressed binary implementation, these conditional statements are replaced by a set of bitwise operations, as described above (Fig. 11). In addition, the CPU implementation calculates a bound on the maximum possible value for F^i for a given combination i. If this value is less than the intermediate value for F_{max} (Eq. (1)), subsequent processing for the combination is skipped. Although this strategy improved performance on the CPU, eliminating this branch and bound logic on the GPU resulted in an additional 6% average speedup for the 2-hit algorithm and 3-hit algorithm (Supplementary Tables S3 and S5).

The multi-hit algorithm only considers combinations represented by the upper triangular matrix, i.e. combinations of g_i and g_j where $i < j$. In the CPU implementation, processing is limited to the upper triangular matrix by loop control conditions. To eliminate these conditional branches, using the formulation from ref. [83], modified

for upper triangular matrix instead of lower triangular matrix, we map the thread index λ to the upper triangular matrix $i < j$ as follows:

$$\begin{aligned} j &= \lfloor \sqrt{1/4 + 2\lambda} + 1/2 \rfloor \\ i &= \lambda - j(j-1)/2 \end{aligned} \tag{2}$$

Using shared memory for parallel reduction. The F^i_{max} values calculated by each GPU thread i is stored in global memory (Fig. 12). The maximum value F_{max} across all these threads can be calculated using parallel reduction, directly in global memory. However, accessing global memory is significantly slower than accessing shared memory[44]. Therefore, we divide global memory data into blocks which are copied into shared memory. Parallel reduction is performed within each block using shared memory to compute the F^j_{max} for block j. The result is copied back to a new allocation in global memory. This new allocation is 1024 (the number of virtual threads per block) times smaller than the original allocation. This process is repeated with the newly allocated values until the single F_{max} values has been calculated. The above approach reduced the total global memory used by approximately 50%, e.g. 2.87 GB for BRCA compared to 5.75 GB without this approach.

Mutation data. Input data for the algorithm consists of two Gene-Sample Mutation matrices, one for tumor samples and another for normal samples. Each element M_{ij} of the matrix is either 1 or 0 depending on if gene i has a protein altering (missense, nonsense, insertion or deletion) somatic mutation in sample j, or not. This information was calculated from whole exome sequencing data available from The Cancer Genome Atlas (TCGA) database[84]. Somatic mutation data, calculated using Mutect2[85], for matched tumor and blood derived normal samples was available for download from TCGA in Mutation Annotation Format (MAF). For 333 normal samples with matched blood derived normal samples, we calculated somatic mutations using the same Mutect2 protocol. See ref. [41] for additional details.

Speedup calculation. Speedup was calculated as t_{ref}/t_{new} where t_{new} and t_{ref} are the run times for the new code and the baseline reference, respectively. Run time was determined using the Linux `time` command, with run time = `sys time` + `user time`. For identifying 2-hit combinations, the run time for the original CPU implementation[41] was used as the baseline reference t_{ref}. However, this was not practical for 3-hit combinations for cancer types where the run time using the original matrix code was over 30 days. Therefore we estimated the run time for cancer types taking over 30 days, based on the actual run times for cancer types requiring less than 30 days. We assumed that the average ratio of 2-hit vs. 3-hit speedup for the compressed binary CPU implementation compared to the original matrix CPU implementation is the same for both categories (run time <30 days and run time >30 days). For cancer types with run time <30 days, we calculated the average ratio $R = \text{Avg}(S_3/S_2)$, where S_3 is the 3-hit speedup and S_2 is the 2-hit speedup, for the CPU compressed binary implementation compared to the CPU original matrix implementation. For cancer types with 3-hit run time >30 days, we estimated the run time as $R \cdot S_2 \cdot t_{3cb}$, where t_{3cb} is the 3-hit run time for the compressed binary CPU implementation. See Supplementary Table S4 for a list of actual and estimated 3-hit run times.

Accuracy calculation. All available mutation data was randomly partitioned into two subsets, with 75% of the data (Training set) used to identify the multi-hit combination using the above algorithm. The remaining data (Test set) was used to calculate the sensitivity, specificity, and 95% confidence interval for the identified set of combinations' ability to differentiate between tumor and normal samples. Sensitivity was calculated as TP/N_t, where TP is the number of true positives (number of tumor samples containing one of the identified combinations) and N_t is the number of tumor samples. Specificity was calculated as TN/N_n, where TN is the number of true negatives (number of normal samples without any of the identified combinations), and N_n is the number of normal samples. 95% confidence interval was calculated using the "exact" Clopper-Pearson method[86]. Overall sensitivity and specificity are calculated from the total count of true positives, true negatives, tumor samples, and normal samples for all cancer types, using the randomly selected 25% test data set. However, it is important to keep in mind that these multi-hit gene combinations may not represent cancer genes. Additional experimental validation is required to determine if mutations within these genes may play a role in cancer.

Data availability

All source code for the multi-hit algorithm are available for download from https://bitbucket.org/qaisalhajri/multihit-gpu-implementation/src/master/.

Received: 29 August 2019; Accepted: 20 January 2020;
Published online: 06 February 2020

References

1. Siegel, R. L., Miller, K. D. & Jemal, A. Cancer statistics, 2019. *CA: A Cancer Journal for Clinicians* **69**, 7–34 (2019).
2. Colditz, G. A., Wolin, K. Y. & Gehlert, S. Applying what we know to accelerate cancer prevention. *Science Translational Medicine* **4**, 127rv4–127rv4 (2012).
3. Maeda, H. & Khatami, M. Analyses of repeated failures in cancer therapy for solid tumors: poor tumor-selective drug delivery, low therapeutic efficacy and unsustainable costs. *Clin. Transl. Medicine* **7**, 11 (2018).
4. Kuchenbaecker, K. B. *et al*. Risks of breast, ovarian, and contralateral breast cancer for BRCA1 and BRCA2 mutation carriers. *JAMA* **317**, 2402–2416 (2017).

5. Jasperson, K. W., Patel, S. G. & Ahnen, D. J. APC-associated polyposis conditions. In *GeneReviews[Internet]* (University of Washington, Seattle, 2017).
6. Pantziarka, P. Primed for cancer: Li Fraumeni Syndrome and the pre-cancerous niche. *ecancermedicalscience* **9**, 541 (2015).
7. Guha, T. & Malkin, D. Inherited TP53 mutations and the Li-Fraumeni syndrome. *Cold Spring Harb Perspect Med* **7**, a026187 (2017).
8. Amadou, A., Waddington Achatz, M. & Hainaut, P. Revisiting tumor patterns and penetrance in germline TP53 mutation carriers: temporal phases of Li-Fraumeni syndrome. *Curr Opin Oncol* **30**, 23–29 (2018).
9. Knudson, A. G. Mutation and cancer: statistical study of retinoblastoma. *Proceedings of the National Academy of Sciences* **68**, 820–823 (1971).
10. Al-Lazikani, B., Banerji, U. & Workman, P. Combinatorial drug therapy for cancer in the post-genomic era. *Nature Biotechnology* **30**, 679 (2012).
11. Ledford, H. Cocktails for cancer with a measure of immunotherapy. *Nature* **532**, 162–164 (2016).
12. Stahl, M. *et al*. Epigenetics in Cancer: A hematological perspective. *PLoS Genet* **12**, e1006193 (2016).
13. Schneider, G., Rad, R., Saur, D. & Schmidt-Supprian, M. Tissue-specific tumorigenesis: context matters. *Nat Rev Cancer* **17**, 239–53 (2017).
14. Almassalha, L. *et al*. The greater genomic landscape: The heterogeneous evolution of cancer. *Cancer Res* **76**, 5605–9 (2016).
15. Vogelstein, B. *et al*. Cancer genome landscapes. *Science* **339**, 1546–58 (2013).
16. Martincorena, I. *et al*. High burden and pervasive positive selection of somatic mutations in normal human skin. *Science* **348**, 880–886 (2015).
17. Martincorena, I. *et al*. Somatic mutant clones colonize the human esophagus with age. *Science* **362**, 911–917 (2018).
18. Lawrence, M. S. *et al*. Mutational heterogeneity in cancer and the search for new cancer-associated genes. *Nature* **499**, 214 (2013).
19. Tian, R., Basu, M. & Capriotti, E. Contrastrank: a new method for ranking putative cancer driver genes and classification of tumor samples. *Bioinformatics* **30**, 572–578 (2014).
20. Tamborero, D., Gonzalez-Perez, A. & Lopez-Bigas, N. Oncodriveclust: exploiting the positional clustering of somatic mutations to identify cancer genes. *Bioinformatics* **29**, 2238–2242 (2013).
21. Dees, N. D. *et al*. Music: identifying mutational significance in cancer genomes. *Genome Res* **22**, 1589–1598 (2012).
22. Kumar, R. D., Swamidass, S. J. & Bose, R. Unsupervised detection of cancer driver mutations with parsimony-guided learning. *Nat Genet* **48**, 1288–1294 (2016).
23. Cheng, F., Zhao, J. & Zhao, Z. Advances in computational approaches for prioritizing driver mutations and significantly mutated genes in cancer genomes. *Briefings in Bioinformatics* **17**, 642–656 (2015).
24. Xi, J., Wang, M. & Li, A. Discovering mutated driver genes through a robust and sparse co-regularized matrix factorization framework with prior information from mRNA expression patterns and interaction network. *BMC Bioinformatics* **19**, 1–14 (2018).
25. Pon, J. R. & Marra, M. A. Driver and passenger mutations in cancer. *Annual Review of Pathology: Mechanisms of Disease* **10**, 25–50 (2015).
26. Lawrence, M. S. *et al*. Discovery and saturation analysis of cancer genes across 21 tumour types. *Nature* **505**, 495 (2014).
27. Bailey, M. H. *et al*. Comprehensive characterization of cancer driver genes and mutations. *Cell* **173**, 371–385 (2018).
28. Merid, S. K., Goranskaya, D. & Alexeyenko, A. Distinguishing between driver and passenger mutations in individual cancer genomes by network enrichment analysis. *BMC Bioinformatics* **14**, 308 (2014).
29. Leiserson, M. D., Reyna, M. A. & Raphael, B. J. A weighted exact test for mutually exclusive mutations in cancer. *Bioinformatics* **32**, 736–745 (2016).
30. Anandakrishnan, R., Varghese, R. T., Kinney, N. A. & Garner, H. R. Estimating the number of genetic mutations (hits) required for carcinogenesis based on the distribution of somatic mutations. *PLoS Comput Biol* **15**, e1006881 (2019).
31. Tomasetti, C., Marchionni, L., Nowak, M. A., Parmigiani, G. & Vogelstein, B. Only three driver gene mutations are required for the development of lung and colorectal cancers. *Proc Natl Acad Sci USA* **112**, 118–123 (2015).
32. Zhang, X. & Simon, R. Estimating the number of rate limiting genomic changes for human breast cancer. *Breast Cancer Res Treat* **91**, 121–124 (2005).
33. Luebeck, E. G. & Moolgavkar, S. H. Multistage carcinogenesis and the incidence of colorectal cancer. *Proc Natl Acad Sci USA* **99**, 15095–15100 (2002).
34. Little, M. & Wright, E. A stochastic carcinogenesis model incorporating genomic instability fitted to colon cancer data. *Mathematical Biosciences* **183**, 111–134 (2003).
35. Ashley, D. The two "hit" and multiple "hit" theories of carcinogenesis. *Br J Cancer* **23**, 313 (1969).
36. Armitage, P. & Doll, R. The age distribution of cancer and a multi-stage theory of carcinogenesis. *Br J Cancer* **8**, 1 (1954).
37. Nordling, C. A new theory on the cancer-inducing mechanism. *Br J Cancer* **7**, 68 (1953).
38. Pires, M. M., Hopkins, B. D., Saal, L. H. & Parsons, R. E. Alterations of EGFR, p53 and PTEN that mimic changes found in basal-like breast cancer promote transformation of human mammary epithelial cells. *Cancer biology & therapy* **14**, 246–253 (2013).
39. Usha, L., Dewdney, S. B. & Buckingham, L. E. Tumor screening and DNA testing in the diagnosis of Lynch syndrome. *JAMA* **316**, 93–94 (2016).
40. MacPherson, D. & Dyer, M. A. Retinoblastoma: From the two-hit hypothesis to targeted chemotherapy. *Cancer Research* **67**, 7547–7550 (2007).
41. Dash, S. *et al*. Differentiating between cancer and normal tissue samples using multi-hit combinations of genetic mutations. *Scientific Reports* **9**, 1005 (2019).
42. NVIDIA Tesla V100 GPU Architecture: The world's most advanced datacenter GPU. Tech. Rep., NVIDIA, Also available at https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf (2017).
43. Jain, T. & Agrawal, T. The haswell microarchitecture-4th generation processor. *International Journal of Computer Science and Information Technologies* **4**, 477–480 (2013).
44. Jia, Z., Maggioni, M., Staiger, B. & Scarpazza, D. P. Dissecting the nvidia volta gpu architecture via microbenchmarking. *arXiv preprint arXiv:1804.06826* (2018).
45. Intel. Product specifications: Intel Xeon Processor E5-2630 v4, https://ark.intel.com/content/www/us/en/ark/products/92981/intel-xeon-processor-e5-2630-v4-25m-cache-2-20-ghz.html, Accessed 2019-12-30 (2017).
46. NVIDIA. Cuda C++ Best Practices Guide, https://docs.nvidia.com/cuda/cuda-c-best-practices-guide/index.html#instruction-optimization, Accessed 2019-12-30 (2019).
47. Anandakrishnan, R. *et al*. Accelerating electrostatic surface potential calculation with multi-scale approximation on graphics processing units. *Journal of Molecular Graphics and Modelling* **28**, 904–910 (2010).
48. Sarbazi-Azad, H. *Advances in GPU Research and Practice: A volume in Emerging Trends in Computer Science and Applied Computing*, chap. 23, 649–705 (Morgan Kaufmann, 2017).
49. Bertil Schmidt, M. S. C. H. Jorge González-Domínguez. *Parallel Programming: Concepts and Practice*, 225–285 (Morgan Kaufmann, 2018).
50. Sarbazi-Azad, H. *Advances in GPU Research and Practice: A volume in Emerging Trends in Computer Science and Applied Computing*, chap. 9, 543–580 (Morgan Kaufmann, 2017).
51. Ahmed, A. A. *et al*. Driver mutations in TP53 are ubiquitous in high grade serous carcinoma of the ovary. *The Journal of Pathology* **221**, 49–56 (2010).

52. Schildkraut, J. M. *et al*. Single nucleotide polymorphisms in the TP53 region and susceptibility to invasive epithelial ovarian cancer. *Cancer Research* **69**, 2349–2357 (2009).
53. Eliopoulos, A. G. *et al*. The control of apoptosis and drug resistance in ovarian cancer: influence of p53 and Bcl-2. *Oncogene* **11**, 1217–1228 (1995).
54. Harłiozińska, A. & Bar, J. K. Relationship between p53 and c-erbB-2 overexpression in tissue sections and cyst fluid cells of patients with ovarian cancer. *Tumor Biology* **15**, 223–229 (1994).
55. Goff, B. *et al*. Overexpression and relationships of HER-2/neu, epidermal growth factor receptor, p53, Ki-67, and tumor necrosis factor alpha in epithelial ovarian cancer. *European Journal of Gynaecological Oncology* **17**, 487–492 (1996).
56. Elbendary, A. A. *et al*. Relationship between p21 expression and mutation of the p53 tumor suppressor gene in normal and malignant ovarian epithelial cells. *Clinical Cancer Research* **2**, 1571–1575 (1996).
57. Song, H., Hollstein, M. & Xu, Y. p53 gain-of-function cancer mutants induce genetic instability by inactivating atm. *Nature Cell Biology* **9**, 573 (2007).
58. Liu, D., Song, H. & Xu, Y. A common gain of function of p53 cancer mutants in inducing genetic instability. *Oncogene* **29**, 949 (2010).
59. Chen, S. *et al*. Gain-of-function mutant p53 enhances hematopoietic stem cell self-renewal. *Blood* **124**, 260 (2014).
60. Wang, H.-Y. *et al*. Role of KCNB1 in the prognosis of gliomas and autophagy modulation. *Scientific Reports* **7**, 14 (2017).
61. Marini, C. *et al*. Clinical features and outcome of 6 new patients carrying de novo KCNB1 gene mutations. *Neurology Genetics* **3**, e206 (2017).
62. Miao, P. *et al*. Genotype and phenotype analysis using an epilepsy-associated gene panel in Chinese pediatric epilepsy patients. *Clinical Genetics* **94**, 512–520 (2018).
63. Calhoun, J. D., Vanoye, C. G., Kok, F., George, A. L. & Kearney, J. A. Characterization of a KCNB1 variant associated with autism, intellectual disability, and epilepsy. *Neurology Genetics* **3**, e198 (2017).
64. Latypova, X. *et al*. Novel kcnb1 mutation associated with non-syndromic intellectual disability. *Journal of Human Genetics* **62**, 569 (2017).
65. Thiffault, I. *et al*. A novel epileptic encephalopathy mutation in KCNB1 disrupts Kv2.1 ion selectivity, expression, and localization. *Journal of General Physiology* **146**, 399–410 (2015).
66. Saitsu, H. *et al*. De novo KCNB1 mutations in infantile epilepsy inhibit repetitive neuronal firing. *Scientific Reports* **5**, 15199 (2015).
67. Deng, Y. *et al*. Slow skeletal muscle troponin t, titin and myosin light chain 3 are candidate prognostic biomarkers for Ewing's sarcoma. *Oncology Letters* **18**, 6431–6442 (2019).
68. Khan, A. *et al*. Homozygous missense variant in the TTN gene causing autosomal recessive limb-girdle muscular dystrophy type 10. *BMC Medical Genetics* **20**, 166 (2019).
69. Yu, M. *et al*. Novel TTN mutations and muscle imaging characteristics in congenital titinopathy. *Annals of Clinical and Translational Neurology* (2019).
70. Jang, J. Y., Park, Y., Jang, D.-H., Jang, J.-H. & Ryu, J. S. Two novel mutations in TTN of a patient with congenital myopathy: A case report. *Molecular Genetics & Genomic Medicine* (2019).
71. Corden, B. *et al*. Association of Titin-truncating genetic variants with life-threatening cardiac arrhythmias in patients with dilated cardiomyopathy and implanted defibrillators. *JAMA Network Open* **2**, e196520–e196520 (2019).
72. Kellermayer, D., Smith, J. E. & Granzier, H. Titin mutations and muscle disease. *Pflügers Archiv-European Journal of Physiology* **471**, 673–682 (2019).
73. Spencer, D. *et al*. Performance of common analysis methods for detecting low-frequency single nucleotide variants in targeted next-generation sequence data. *J Mol Diag* **16**, 75–88 (2014).
74. Sandmann, S. *et al*. Evaluating variant calling tools for non-matched next-generation sequencing data. *Sci Rep* **7**, 43169 (2017).
75. Cerami, E. *et al*. The cBio cancer genomics portal: An open platform for exploring multidimensional cancer genomics data. *Cancer Discovery* **2**, 401–404 (2012).
76. Goldschmidt, O., Hochbaum, D. S. & Yu, G. A modified greedy heuristic for the set covering problem with improved worst case bound. *Information Processing Letters* **48**, 305–310 (1993).
77. Crescenzi, P., Kann, V., Halldórsson, M. & Karpinski, M. A compendium of NP optimization problems, https://www.nada.kth.se/~viggo/problemlist/compendium.html, Accessed 2019-12-27 (1995).
78. Hartmanis, J. Computers and intractability: a guide to the theory of NP-completeness. *Siam Review* **24**, 90 (1982).
79. Anandakrishnan, R. A partition function approximation using elementary symmetric functions. *PloS One* **7**, e51352 (2012).
80. Kernighan, B. & Ritchie, D. M. *The C programming language* (Prentice hall, 2017).
81. Harris, M. Optimizing parallel reduction in CUDA, https://developer.download.nvidia.com/assets/cuda/files/reduction.pdf, Accessed 2019-12-27 (2019).
82. Singh, I., Shriraman, A., Fung, W. W., O'Connor, M. & Aamodt, T. M. Cache coherence for gpu architectures. In *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, 578–590 (IEEE, 2013).
83. Navarro, C. A. & Hitschfeld, N. Gpu maps for the space of computation in triangular domain problems. In *2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC, CSS, ICESS)*, 375–382 (IEEE, 2014).
84. Weinstein, J. *et al*. The cancer genome atlas pan-cancer analysis project. *Nat Genet* **48**, 1288–1294 (2016).
85. do Valle, Í. F. *et al*. Optimized pipeline of mutect and gatk tools to improve the detection of somatic single nucleotide polymorphisms in whole-exome sequencing data. *BMC Bioinformatics* **17**, 341 (2016).
86. Clopper, C. J. & Pearson, E. S. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika* 404–413 (1934).
87. Zhang, H., Meltzer, P. & Davis, S. Rcircos: an R package for Circos 2D track plots. *BMC Bioinformatics* **14**, 244 (2013).
88. Guo, X. g3viz: Interactively Visualize Genetic Mutation Data using a Lollipop-Diagram, https://github.com/G3viz/g3viz, Accessed 2019-12-27 (2019).

Acknowledgements

This work was supported by a VCOM Bradly Foundation grant and a VCOM REAP grant.

Author contributions

Q.H., S.D. and R.A. designed the algorithm, implemented the software, analyzed the results and wrote the manuscript. R.A., W.F. and H.G. contributed to the research design and manuscript review.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information is available for this paper at https://doi.org/10.1038/s41598-020-58785-y.

Correspondence and requests for materials should be addressed to R.A.