İkiden Fazla Tensörün En Verimli Daraltma Sırası Optimal Contraction Order of Multiple Tensors

Can Kavaklıoğlu, A. Taylan Cemgil
Bilgisayar Mühendisliği Bölümü
Boğaziçi Üniversitesi
İstanbul, Türkiye
can.kavaklioglu, taylan.cemgil@boun.edu.tr

***Özetçe* —Büyük veri içeren hesaplama problemlerini çözmek için çok bağıntılı analiz yöntemlerini güncel paralel donanımlar üzerinde uygulamak gerekmektedir. Tensör daraltılması işlemi çok bağıntılı analizinde kullanılan temel işlemler arasındadır. Sunduğumuz dinamik programlama formülasyonu birden fazla tensörün daraltılması işlemindeki en az hafıza gerektiren daraltma sırasını tespit etmektedir.**

Anahtar Kelimeler—Tensör daraltması, dinamik programlama

***Abstract*—Computational problems involving big data require application of multiway analysis tools on modern parallel infrastructure. Tensor contraction is one the fundemental operations used in multiway analysis. Dynamic programming formulation presented in this work searches for the least memory using contraction order in multiple tensor contraction operation.**

Keywords—Tensor contraction, dynamic programming

I. GİRİŞ

Güncel hesaplama problemlerinin çözülmesi için işlenmesi gereken veri miktarı her geçen gün artmaktadır. Artan veri miktarı ile işlenen veri içerisindeki tespit edilmesi gereken ilişkiler de yoğunlaşır. Büyük miktardaki yoğun ilişkili verinin analizi için en uygun yöntemlerden birisi çok bağıntılı (Multiway) veri analizdir. Çok bağıntılı analizde temel olarak, n sayıda bağımsız indisi olan dizi olarak tanımladığımız tensör veri yapıları kullanılır. Veri analizinde tensör veri yapılarının kullanılmasıyla hem boyut azaltma teknikleri ile ortaya çıkan veri kaybının önüne geçilebilir hem de güncel paralel veri işleme tekniklerinden azami düzeyde faydalanma imkanı ortaya çıkar.

Tensör veri yapısı farklı literatürlerdeki çalışmalarda karşımıza çıkmaktadır. Örüntü işleme [1], sinyal işleme [2], nöroloji [3] alanlarındaki çalışmalar çok bağıntılı analizin farklı araştırma dallarında kullanılmaya başlandığını göstermektedir [4].

Kuvantum kimya hesaplamalarında kullanılan çok bağıntılı denklemleri otomatik olarak sadeleştirdikten sonra paralel sistemler üzerinde çalışacak kod üretecek çalışmalar yapılmıştır [5]. Bu çalışmalar sayesinde bilim insanlarının uzun çalışmalar sonucunda üretebileceği denklemler bilgisayarlar tarafından daha hızlı bir şekilde üretilebilmektedir. Otomatik kod üretme yöntemi temel veri yapısı olarak tensörleri kullanarak verilen problem ve sistem bilgilerine göre en uygun işlem sırasını aramaktadır.

978-1-4673-5563-6/13/$31.00

Araştırma grubumuzun çalışmalarından Saklı Tensör Ayrıştırma (STA) çerçevesi tensör veri yapılarını çok bağıntılı ayrıştırma problemlerinde saklı tensörleri hesaplamak için kullanır [6]. Çerçevenin güncelleme yönteminin en temel işlemi birden fazla tensörün daraltılması işlemidir.

Bahsedilen hesaplama yöntemlerini araştırmacılar için ilginç kılan en temel ortak özellik kullanılması gereken büyük veri miktarıdır. Büyük miktarda veri kullanarak hesaplama yapabilmek için kullanılacak yöntemlerin en temel aşamalarından itibaren kullanılacak sistemin özellikleri de göz önüne alınarak dikkatle tasarlanması gerekmektedir.

Tensör veri yapısını kullanan yöntemlerin en temel işlemlerden birisi iki ya da daha fazla sayıda tensörün daraltılması işlemidir. İkiden fazla matrisin çarpımı işlemindeki birleşirlik kuralını kullanarak en az işlem gerektiren çarpım sırasını seçilmesine benzer bir şekilde ikiden fazla tensörün daraltılması operasyonunda da işlem sırasını değiştirerek aynı sonuca farklı sayıda işlem yaparak varan sonuçlar bulmak mümkündür. Bu çalışmada ikiden fazla tensörün daraltılması işlemindeki en verimli işlem sırasını tespit eden algoritmamızı sunacağız.

II. TENSÖR VERİ YAPISI

Tensör veri yapısını n sayıda bağımsız indisi olan dizi olarak tanımlıyoruz. Tek indise sahip tensörü tek boyutta elemanı olan bir vektör, iki indise sahip tensörü matris olarak düşünebiliriz. Makale içerisinde boyut ifadesi indis sayısını ifade etmek için kullanılmıştır.

Hesaplama problemini tanımlamak için kullanılacak bir tensörün yapısını belirtirken iki bilginin verilmesi gereklidir, tensörün adı ve kullandığı indislerin isimleri. Kullandığımız notasyonda tensörün ismi büyük harf ile, tensörün indisleri ise parantez içerisinde kullanılan indislerin isimleri yazılarak ifade edilmektedir. Örnek olarak sadece i boyutunda elemanı olan A ismindeki tensörü belirtmek için $A(i)$ ifadesini hem i hem de j boyutunda elemanı olan B tensörünü belirtmek için $B(i,j)$ ifadesini kullanabiliriz.

Tek bir küçük harf ile ifade ettiğimiz boyut tanımlarının büyüklükleri de hesaplamalar için önemlidir. i boyutunun en fazla kaç farklı değer ifade edebileceği $|i| = 5$ şeklinde belirtilir. Boyut büyüklükleri bir problem tanımı için sabit olarak kabul edilir. Farklı tensörlerde bulunan aynı isimli boyut tanımlarının aynı büyüklükte olmasını bekleriz.

Tensör veri yapılarını kullanarak bir matris çarpımı işlemi ifade etmek istersek, aşağıdaki gibi bir problem tanımı belir-

tebiliriz:

$$F(i,k,j) = A(i,k)\ B(k,j)$$
$$X(i,j) = \sum_{k} F(i,k,j) \tag{1}$$

İlk aşamada $A(i,k)$ ve $B(k,j)$ girdi tensörlerinin ilgili elemanları çarpılır ve sonuça oluşacak $F(i,k,j)$ tensörü k indisi üzerinden daraltılarak $X(i,j)$ çıktı tensörü hesaplanır.

Diğer bir bakış açısıyla matris çarpımındaki içler çarpımı işlemi tensör daraltması işlemi kullanılarak gerçekleştirilmiştir. Aşağıdaki bölümlerde tensör daraltması işlemininin detayları ve en az hafıza kullanan işlem sırasının tespit edilmesi konularından bahsedilecektir.

III. BİR ÇİFT TENSÖRÜN DARALTILMASI

Bir çift tensörün daraltılması işlemi tensör veri yapısını kullanan hesaplama yöntemlerinin temel işlemlerindendir. Bu işlemde $Z_1(v_1)$ ve $Z_2(v_2)$ tensörleri girdi olarak alınır ve $X(v_0)$ çıktı tensörü hesaplanır. Gerçekleşen işlemin karakteristikleri girdi ve çıktı tensörlerinin indis kümeleri $\{v_0, v_1, v_2\} \in V$ tarafından tanımlanır.

Bir çift tensörün daraltılması işlemini iki adımda gerçekleştirebiliriz. İlk adımda $Z_1(v_1)$ ve $Z_2(v_2)$ girdi tensörleri ilgili v_1 ve v_2 indis kümeleri üzerinden çarpılarak $F(v_3)$ ara tensörü oluşturulur:

$$F(v_3) = Z_1(v_1)\ Z_2(v_2) \qquad v_3 = \{v_1 \cup v_2\} \tag{2}$$

İkinci adımda ara tensörün indis kümesi v_3'te bulunan ancak çıktı tensörünün indis kümesi v_0'da bulunmayan indisler $\bar{v}_0 = \{v_3 \setminus v_0\}$ üzerinden daraltma işlemi gerçekleştirilir:

$$X(v_0) = \sum_{\bar{v}_0} F(v_3) \tag{3}$$

İki adımı beraber tek satırda aşağıdaki gibi ifade edebiliriz:

$$X(v_0) = \sum_{\bar{v}_0} Z_1(v_1)\ Z_2(v_2) \tag{4}$$

Örnek olarak indis kümelerini $v_0 = \{i,j\}$, $v_1 = \{i,k\}$, $v_2 = \{k,j\}$ şeklinde tanımlarsak matris çarpımı işlemini bir çift tensörün daraltılması işlemi olarak belirtmiş oluruz:

$$X(i,j) = \sum_{k} Z_1(i,k)\ Z_2(k,j) \qquad \{i,j,k\} \in V \tag{5}$$

V indis kümesi hesaplama probleminde kullanılan bütün boyutları belirtmektedirler.

Altenatif bir hesaplama yönteminde F ara tensörünü hesaplamadan, direk olarak X çıktı tensörünün her bir elemanı için işlem yaparak aynı sonuca ulaşmak mümkündür. Bu durumda F tensörünün saklanması için gereken hafıza miktarından tasarruf edilir. Karşılık olarak X tensörünün her bir elemanını hesaplamak için kullanılacak girdi tensörü elemanlarını tespit etme işlemi için fazladan hesaplama gereksinimi ortaya çıkmaktadır. Uygulamada hangi alternatifin kullanılacağı kullanılacak girdi/çıktı tensörlerin indis kümesi yapısı ve donanım kaynaklara göre seçilmelidir.

IV. İKİDEN FAZLA TENSÖRÜN DARALTILMASI

Bir çok veri analizi probleminde ikiden fazla tensörün daraltılması operasyonuna gerek duyulmaktadır. Bir çift tensörün daraltılmasından farklı olarak girdi tensörü sayısı sınırlı değildir. Denklem 6'da tanımlanan işlemi Algoritma 1'deki gibi, indis kümesi $v_0 = \cup_n v_n$ olan tek bir F ara tensörü kullanarak gerçekleştirmek mümkündür.

$$X(v_0) = \sum_{\bar{v}_0} \prod_{n} Z_n(v_n) \tag{6}$$

Algoritma 1 İkiden fazla tensörün daraltılması (tek ara tensör)

$F(v_0) = Z_1(v_1)Z_2(v_2)$
for $i = 3 : n$ **do**
  $F(v_0) = F(v_0)Z_i(v_i)$
end for

Eğer hesaplama için kullanılacak donanımda $\cup_n v_n$ indis kümesini saklayacak kadar hafıza mevcutsa bu yöntemi kullanmak paralleştirme açısından en verimli yöntem olduğu için tercih edilebilir. Ancak büyük veri kullanılan problemlerde $\cup_n v_n$ indis kümesini saklayabilecek büyüklükte hafızanın sistemde mevcut olması ender karşılaşılan bir durumdur.

Gerekli hafıza miktarı açısından daha makul gereksinimleri olan diğer bir çözüm eldeki donanımın hafıza kapasitesine sığacak büyüklükte birden fazla ara tensör kullanmaktır. Örnek olarak birden fazla tensörün daraltılmasını gerektiren problemlerden birisi olan Tucker3 modelinde $X(i,j,k) = \sum_{p,q,r} A(i,p)B(j,q)C(k,r)G(p,q,r)$ birden fazla ara tensör kullanarak altı farklı yoldan aynı sonuca ulaşmak mümkündür. Seçeneklerden iki tanesi aşağıda verilmiştir:

Birinci daraltma sırası seçeneği:

$$\begin{aligned} CG(k,p,q) &= \sum_{r} C(k,r)\ G(p,q,r) && \text{daraltma: r} \\ BCG(j,k,p) &= \sum_{q} B(j,q)\ CG(k,p,q) && \text{daraltma: q} \\ X(i,j,k) &= \sum_{p} A(i,p)\ BCG(j,k,p) && \text{daraltma: p} \end{aligned} \tag{7}$$

İkinci daraltma sırası seçeneği:

$$\begin{aligned} AG(i,q,r) &= \sum_{p} A(i,p)\ G(p,q,r) && \text{daraltma: p} \\ CAG(k,i,q) &= \sum_{r} C(k,r)\ AG(i,q,r) && \text{daraltma: r} \\ X(i,j,k) &= \sum_{q} B(j,q)\ CAG(k,i,q) && \text{daraltma: q} \end{aligned} \tag{8}$$

Birinci işlem sırasında ara tensör hafızası ihtiyacı $|k|*|p|*|q| + |j|*|k|*|p|$ büyüklüğünde, ikincisinde ise $|i|*|q|*|r| + |k|*|i|*|q|$ büyüklüğündedir. X çıktı tensörünün hafıza gereksinimi bütün olası işlem sıralarında aynı olduğu için dikkate alınmamaktadır. Bu örneklerden de açıkca görülmektedir ki farklı daraltma sıraları seçildiğinde farklı miktarlarda ara tensör hafızası miktarı ihtiyacı ortaya çıkmaktadır.

V. EN VERİMLİ DARALTMA SIRASI

Tensör daraltması işlemi bir çok hesaplama operasyonunun en temel işlemi olarak kullanılmaktadır. Dolayısıyla tensör daraltması işleminin en verimli şekilde gerçekleştirilmesi bu operasyonların çözebildiği problemlerin büyüklüğünü arttırma potansiyeline sahiptir.

Örnek olarak Tucker3 ayrışım modelindeki ara tensörlerin yapılarını göz önüne alabiliriz. Bir indis kümesi yapısında olası altı daraltma sırasından sadece birisi sistemdeki hafıza miktarına sığabilecek boyutta, diğer sıraların her birisi sistem hafızasından daha büyük hafıza gereksinimine sahip olabilir. Bu durumda en az hafıza miktarını gerektiren daraltma sırasının tespit edilmesi problemin eldeki donanım ile çözülebilir hale gelmesini sağlayabilir.

En verimli daraltma sırasının bulunması daha önceden farklı açılardan incelenmiş bir konudur [7] [8]. Verilen problemin yapısına ve kullanılacak donanımın özelliklerine göre bir çok farklı yöntem denenebilir. Bunlar arasında cebirsel düzenlemele, paralleştirme miktarının ayarlanması, veri transferine göre yeniden düzenlemele işlemleri sayılabilir. Bu bölümde birden fazla tensörün daraltılma sırasının seçilmesi için geliştirdiğimiz dinamik programlama formülasyonu anlatılacaktır.

A. Dinamik Programlama Formülasyonu

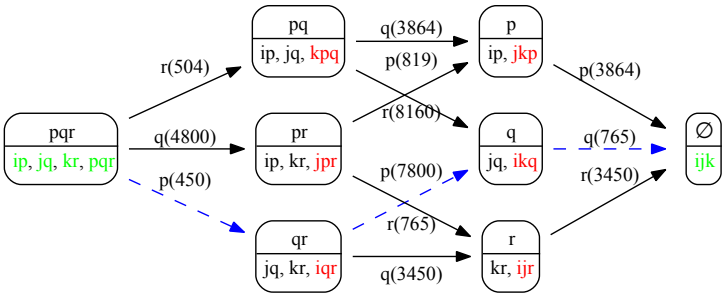
Birden fazla tensörün en verimli işlem sırasının tespiti NP-tam bir arama problemidir [9]. Bununla beraber tek bir çoklu tensör daraltması işleminde en verimli sıranın bulunması pratikte hızlı bir şekilde mümkün olmaktadır. Ancak problemin NP-tam özelliğinden ötürü birden fazla daraltmanın arka arkaya yapılması gerektiği durumlarda arama uzayı üstel bir oranla büyüdüğü için 10-20 arka arkaya yapılan işlemin en verimli işlem sırasının hesaplanması bile saatler sürebilmektedir. Dolayısıyla arka arkaya birden fazla tensörün daraltılması problemi için daha akıllı arama teknikleri ya da yaklaşımsal yöntemler denenmelidir. Bunlar sonraki çalışmalarımızın konuları olacaktır.

Neyseki birden fazla tensörün en verimli daraltma sırasını bulmak için büyük bir arama uzayına ihtiyaç doğmamaktadır. Bununla beraber problemin alt parçaları en verimli şekilde çözüldüğünde büyük problemi de en verimli şekilde çözmek mümkündür. Dolayısıyla en iyi çözüme dinamik programlama (DP) [10] yöntemi ile ulaşabiliriz. Formülasyonu yapabilmek için bir durum tanımına ihtiyaç vardır. Problemimiz için durum tanımı iki elemanlı bir değiken grubu olarak seçilmiştir:

$$D = (\kappa, \rho) \tag{9}$$

κ verilen durumdaki henüz daraltılmamış indisleri, ρ ise verilen durumda mevcut olan faktörleri temsil etmektedir. κ ve ρ bilgileri verildiğinde problemin bir durumunu $D(\kappa, \rho)$ tanımlamak için yeterli bilgi mevcuttur.

Başlangıç durumu $D(\kappa = \bar{v}_0,\ \rho = \{v_1 v_2 \dots v_n\})$ olarak tanımlanır. Daha sonraki durumlarda gerçekleştirilecek daraltma işlemleri ile κ kümesinin elmanları birer birer azalacaktır. ρ başlangıçta sadece $\{v_1 v_2 \dots v_n\}$ girdi tensörleri kullanılarak başlatılır ve sonraki durumlarda girdi tensörler daraltmalar sonucu silinerek, ara tensörler eklenecektir. Daraltma işlemlerinin sonucunda $D(\kappa = \emptyset,\ \rho = \{v_0\})$ durumuna varılır.



Şekil 1. Tucker3 modelinin daraltılmasında kullanılabilecek bütün olası daraltma yollarını gösteren grafik. Her bir düğüm bir $D(\kappa, \rho)$ problem durumunu belirtir. Düğümün üst tarafındaki harfler henüz daraltılmamış $D.\kappa$ daraltma indislerini, alt kısmı ise problemde mevcut olan $D.\rho$ faktörleri ifade eder. Her bir faktör veri sahibi olduğu indislerin isimleri ile belirtilmiştir. Oklar üzerindeki sayılar birikimli hafıza maliyetini belirtir. Hafıza maliyeti okun sonundaki düğümde eklenen ara tensörün hafıza gereksiniminin toplam yol maliyetine eklenmesi olarak tanımlanmıştır. Örnek olarak soldan ilk düğümde p boyutunda daraltma yapıldıktan sonra $AG(i, q, r)$ ara tensörü oluşmuştur ve hafıza gereksinimi $|i| * |q| * |r| = 5 * 9 * 10 = 450$'dir. Dinamik programlama ile çıkarılan maliyet ağacını temsil eden grafik üzerinde yapılacak basit bir arama verilen model için en verimli daraltma sırasını verecektir. Bu model için en verimli daraltma sırası kesik çizgili olarak gösterilmiştir: p, r, q

Bu durumda daraltılması gereken hiç bir indis kalmamıştır ve elde edilen indis kümesi çıktı tensörün indis kümesine eşittir.

Kullanılan hafıza miktarını maliyet olarak tanımlayan DP formülasyonu şu şekilde belirtilir:

$$\begin{aligned}\text{maliyet}(D) = \min_b \{&\text{hafıza\_kullanımı}(D) + \\ &\text{maliyet}(D');\ \ b \in D.\kappa\}\end{aligned} \tag{10}$$

Bir problemden diğer bir alt probleme geçiş yapmak için her hangi bir d boyutunda daraltma işlemi yapılması gerekir:

$$D \downarrow_d = D' = (D.\kappa \setminus d,\ D.\rho \odot d) \tag{11}$$

$\odot$ ifadesi d boyutunda veriye sahip olan faktörlerin daraltılması işlemini temsil etmektedir.

Dinamik programlamanın doğası gereği önce yukarıda verilen formülasyona göre bir maliyet ağacı üretilir. İkinci adım olarak elde edilen ağaç veri yapısı basit bir açgözlü (greedy) arama sonucunda en verimli daraltma sırası elde edilir [11].

B. Örnek DP çözümü

Algoritmanın indis yapısı $i = 5$, $j = 60$, $k = 7$, $p = 8$, $q = 9$, $r = 10$ olarak tanımlanmış olan Tucker3 ayrıştırma modelini kullanarak ürettiği maliyet ağacı Şekil 1'deki grafikte gösterilmiştir. Şekildeki grafiğin her bir düğümü dinamik programlama sürecinin bir durumunu $D(\kappa, \rho)$ gösterir. Düğümün üst kısımı problem durumunda henüz daraltılmamış $D.\kappa$ daraltma indislerini gösterir. Alt kısım ise durumda bulunan $D.\rho$ faktörlerinin indislerini kullanarak gösterir.

Verilen problem kullanılarak oluşturulabilecek her bir durumda daraltılması gereken her bir boyut için ilgili düğümden çıkan bir ok çizilmiştir. Okun etiketi seçilen daraltma indisini ve daraltma esnasında ihtiyaç duyulacak ara tensörün hafıza gereksinimi parantez içerisindeki sayı ile belirtir. Örnek olarak şekildeki soldan ilk düğümden çıkan oklardan en alttakinin etiketi $p(450)$ olarak belirtilmiştir. Bu ifade p boyutunda

yapılacak daraltılma sonucunda oluşacak $AG(i,q,r)$ ara tensörünün hafıza gereksinimi belirtmektedir $|i| * |q| * |r| = 5 * 9 * 10 = 450$.

Şekildeki grafik oluşturulduktan sonra basit bir açgözlü arama ile kesik çizgilerle belirlenmiş olan en verimli daraltma sırası bulunmuştur. Her ne kadar alınacak sonuçlar tamamen indis yapısına bağlı olsa da bu örnekteki problem için %80 verim artışı sağlamak mümkündür.

VI. SONUÇ

Anlatılan işlemlerin pratik olarak gerçekleştirilebilmesi için Tensor Factorization Toolbox (TFT) isminde bir Matlab kütüphanesi (http://www.cmpe.boun.edu.tr/pilab/pilabfiles/pltftoolbox/tft/) gerçellenmektedir. TFT'nin ana hedefi tensör ayrışım modellerini kullanımı kolay bir şekilde Matlab ortamında ifade edebilmek ve tensör ayrışımı tabanlı çıkarım yöntemlerini araştırmacıların kullanımına sunmaktır.

TFT'nin son sürümünde birden fazla tensörün daraltılması işleminin en verimli daraltma sırasını dinamik programlama yöntemi ile tespit edilmesi operasyonu mevcuttur. Araştırmacıların problemlerinde kullandıkları tensör daraltması işlemlerinin en verimli daraltma sırasını tespit ederek daha büyük veri miktarları ile çalışmalarında yardımcı olmaktadır.

İleriki çalışmalarımızda arka arkaya birden fazla tensörün daraltılması işleminde en verimli daraltma sırasını tespit etmek konusunda elde ettiğimiz sonuçları sunacağız. Bununla beraber sunulan algoritmaların ekran kartları üzerinde çalışan paralel sürümleri de yakında TFT arayüzü üzerinden kullanılabilir olacaktır.

TEŞEKKÜR

Bu çalışma Türkiye Bilimsel ve Teknik Araştırmalar Kurumu (TÜBİTAK) tarafından 110E292 nolu araştırma projesi ve Boğaziçi Araştırma Projeleri (BAP) tarafından P5723 ve P6882 nolu araştırma projeleri kapsamında desteklenmektedir.

KAYNAKÇA

[1] M. Vasilescu and D. Terzopoulos, "Multilinear analysis of image ensembles: Tensorfaces," *Computer Vision—ECCV 2002*, pp. 447–460, 2002.

[2] L De Lathauwer, "Fourth-order cumulant-based blind identification of underdetermined mixtures," *IEEE Transactions on Signal Processing*, 2007.

[3] E Martinez-Montes and PA Valdés-Sosa, "Concurrent EEG/fMRI analysis by multiway partial least squares," *NeuroImage*, 2004.

[4] T.G. Kolda and B.W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[5] G. Baumgartner, a. Auer, D.E. Bernholdt, a. Bibireata, V. Choppella, D. Cociorva, R.J. Harrison, S. Hirata, S. Krishnamoorthy, S. Krishnan, M. Nooijen, R.M. Pitzer, J. Ramanujam, P. Sadayappan, and a. Sibiryakov, "Synthesis of High-Performance Parallel Programs for a Class of ab Initio Quantum Chemistry Models," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 276–292, Feb. 2005.

[6] Y Kenan Yılmaz, A Taylan Cemgil, and Umut Şimşekli, "Generalised Coupled Tensor Factorisation," in *Advances in Neural Information Processing Systems*, 2011, pp. 1–9.

[7] Albert Hartono and Alexander Sibiryakov, "Automated operation minimization of tensor contraction expressions in electronic structure calculations," *ICCS*, 2005.

[8] Pai-wei Lai, Huaijian Zhang, Samyam Rajbhandari, Edward Valeev, Karol Kowalski, and P Sadayappan, "Effective Utilization of Tensor Symmetry in Operation Optimization of Tensor Contraction Expressions," in *ICCS 2012*, 2012, vol. 00, pp. 1–10.

[9] CC Lam, "On optimizing a class of multi-dimensional loops with reductions for parallel execution," *Parallel Processing Letters*, 1997.

[10] Richard Bellman, *Dynamic Programming*, Princeton Univ Pr, 1957.

[11] Sanjoy Dasgupta, Christos Papadimitriou, and Umesh Vazirani, *Algorithms*, McGraw-Hill Science, 2006.