Research Article

Roman Kolpakov* and Mikhail Posypkin

Optimality and Complexity Analysis of a Branch-and-Bound Method in Solving Some Instances of the Subset Sum Problem

https://doi.org/10.1515/comp-2020-0212
Received Jul 22, 2019; accepted Aug 24, 2020

Abstract: In this paper we study the question of parallelization of a variant of Branch-and-Bound method for solving of the subset sum problem which is a special case of the Boolean knapsack problem. The following natural approach to the solution of this question is considered. At the first stage one of the processors (control processor) performs some number of algorithm steps of solving a given problem with generating some number of subproblems of the problem. In the second stage the generated subproblems are sent to other processors for solving (one subproblem per processor). Processors solve completely the received subproblems and return their solutions to the control processor which chooses the optimal solution of the initial problem from these solutions. For this approach we define formally a model of parallel computing (frontal parallelization scheme) and the notion of complexity of the frontal scheme. We study the asymptotic behavior of the complexity of the frontal scheme for two special cases of the subset sum problem.

Keywords: computational complexity, parallel computations, subset sum problem, Branch-and-Bound method

1 Introduction

We consider one of the possible parallel realizations of a variant of Branch-and-Bound method for solving the *subset sum problem* which is a particular case of *knapsack problem* [1]. Along with the dynamic programming method [2], the Branch-and-Bound method is a basic method for solving this problem. The Branch-and-Bound method is based on step-by-step decomposition of a given problem to subproblems with removing from the consideration subproblems which certainly have no optimal solutions. The most simple way of parallelization of such computations is as follows. One of the processors being used is chosen as a control processor that processes some number of decompositions on the first stage. On the second stage, the obtained subproblems are sent to the other processors, one subproblem to each processor. Then processors solve completely the received subproblems using the Branch-and-Bound method, and the control processor collects all the obtained solutions and chooses the optimal solution from them. More detailed description of this approach to parallelization of computations can be found in [3]. The described scheme of calculations parallelization suits ideally the distributed environments where the control processor interacts with the other processor only for sending a new job or for receiving calculation results. Many grid systems which are widely used today belong to this class of environments [4].

The subset sum problem (SSP) is a particular case of the knapsack problem where for each item the price is equal to the weight of the item. The subset sum problem is stated as follows:

$$\begin{aligned} &\text{maximize } f(\tilde{x}) = \textstyle\sum_{i\in N} x_i w_i, \\ &\text{subject to } g(\tilde{x}) = \textstyle\sum_{i\in N} x_i w_i \le C, \\ &x_i \in \{0,1\}, i \in N, \end{aligned} \tag{1}$$

where $N = \{1, \dots, n\}$ is a set of integers between 1 and n, a capacity C and weights w_i for $i \in N$ are positive numbers (without loss of generality we assume that $C < \sum_{i\in N} w_i$).

The questions of parallelization of solving of SSP by dynamic programming method [5] and Horowitz and Sahni (two-list) method [6] in computational model with shared memory were actively investigated before in literature. Theoretical bounds on the complexity of parallel solving of SSP are obtained in [7–11] for dynamic programming method and in [12] for two-list method. Empirical parallel algorithms of solving SSP by dynamic programming

***Corresponding Author: Roman Kolpakov:** Lomonosov Moscow State University, Leninskie Gory, Moscow, Russia and Federal Research Center "Computer Science and Control" of RAS, Vavilov st. 40, Moscow, Russia; Email: foroman@mail.ru
Mikhail Posypkin: Moscow Institute of Physics and Technology, Moscow, Russia and Federal Research Center "Computer Science and Control" of RAS, Vavilov st. 40, Moscow, Russia; Email: mposypkin@gmail.com

method for CPU and GPU computational models are proposed in [11, 13, 14]. Empirical parallelizations of two-list method of solving of SSP for GPU and hybrid CPU/GPU computational models are proposed in [15–17]. A parallel GPU algorithm of solving some modification of SSP is proposed in [18]. More information on the state of the art in this field can be found in [19] where an efficient modification of the balanced algorithm [1] for SSP is proposed. New approaches for parallel solving of optimization problems were proposed in [20–24].

In the paper (extended journal version of [25]) we investigate a variant of the parallelization scheme described above which is called *the frontal scheme*. We give a definition of complexity of this scheme and study the asymptotic behavior of this complexity for two series of subset sum problems. Comparing the asymptotical estimates of the frontal scheme complexity obtained for these two series of problems we discover some qualitative effect for this complexity.

2 Preliminaries

A boolean tuple $\tilde{x} = (x_1, x_2, \ldots, x_n)$ such that $g(\tilde{x}) \le C$ is called a *feasible solution* of the problem (1). A feasible solution $\tilde{x}$ of a problem (1) is called an *optimal solution* if for any other feasible solution $\tilde{y}$ of the problem (1) the inequality $f(\tilde{y}) \le f(\tilde{x})$ holds. Solving the problem (1) means finding at least one of its optimal solutions.

We define a *map* as the pair (I, θ) of a set $I \subseteq N$ and a mapping $\theta : I \to \{0, 1\}$. Any map (I, θ) defines a *subproblem* formulated as follows:

$$\begin{aligned} &\text{maximize } f(x) = \textstyle\sum_{i\in N} w_i x_i, \\ &\text{subject to } g(x) = \textstyle\sum_{i\in N} w_i x_i \le C, \\ &x_i = \theta(i), i \in I, \\ &x_i \in \{0, 1\}, i \in N \setminus I. \end{aligned} \tag{2}$$

Variables x_i such that $i \in I$ are called *fixed variables* of the subproblem (2), and variables x_i such that $i \in N \setminus I\}$ are called *free variables* of this subproblem. We will refer to the subproblem (2) as the *corresponding* subproblem for the map (I, θ) and will refer to the map (I, θ) as the *corresponding* map for the subproblem (2). The elements of the map corresponding for a subproblem P will be denoted by I_P and θ_P.

A boolean tuple $\tilde{x} = (x_1, x_2, \ldots, x_n)$ such that

$$\begin{aligned} &g(x) \le C, \\ &x_i = \theta(i), i \in I, \end{aligned}$$

is called *a feasible solution* of the subproblem (2). Clearly, any *feasible solution* of the subproblem (2) is a feasible solution of the problem (1) as well. A feasible solution $\tilde{x}$ of the subproblem (2) is called *optimal* if for any other feasible solution $\tilde{y}$ of this subproblem the inequality $f(\tilde{y}) \le f(\tilde{x})$ holds (for convenience, we will also call the optimal solution of a subproblem simply by solution).

Let $W = \sum_{i\in N} w_i$. We say that the subproblem (2) satisfies *C0-condition* if $\sum_{i\in I} \theta(i) w_i > C$ and satisifies *C1-condition* if $\sum_{i\in I}(1 - \theta(i)) w_i \ge W - C$. For subproblems satisfying C0- and C1-conditions we have the following obvious facts.

Proposition 1. *A subproblem satisfying C0-condition has no feasible solutions.*

Proposition 2. *A subproblem satisfying C1-condition has the only optimal solution $\tilde{x} = (x_1, x_2, \ldots, x_n)$ such that*

$$x_i = \begin{cases} \theta(i) \text{ if } i \in I, \\ 1 \text{ if } i \in N \setminus I. \end{cases}$$

Corollary 1. *A subproblem can not satisfy both C0-condition and C1-condition at the same time.*

Moreover, it is easy to see that in the case of $I = N$ the subproblem (2) satisfies either C0-condition or C1-condition.

Let x_i be a free variable of the subproblem (2). Then we can consider two subproblems P_0 and P_1 such that $I_{P_1} = I_{P_2} = I \cup \{i\}$ and

$$\theta_{P_k}(j) = \begin{cases} \theta(j) \text{ if } j \in I, \\ k \text{ if } j = i, \end{cases} \qquad k = 0, 1.$$

Note that the set of all feasible solutions of the subproblem (2) is the union of the sets of all feasible solutions of the subproblems P_0 and P_1, so any optimal solution of the subproblem (2) is an optimal solution of either the subproblem P_0 or the subproblem P_1. We will say that the subproblem (2) is *decomposed to the subproblems P_0 and P_1 along the variable x_i* and call the subproblem P_0 (P_1) *0-decomposition* (*1-decomposition*) of the subproblem (2). Note that if the subproblem (2) satisfies neither C0-condition nor C1-condition then a 0-decomposition (a 1-decomposition) of this problem can not satisfy C0-condition (C1-condition).

3 Branch-and-Bound algorithm

In this paper we consider one of the basic variants of the Branch-and-Bound method for solving SSP which we call the *consecutive* Branch-and-Bound (cBnB) algorithm. In

order to solve a given subset sum problem by this algorithm, during the procedure of solving we maintain a FIFO subproblems queue containing for each time all subproblems of the given problem which are waiting for processing. To each time of this procedure we also keep the best found feasible solution of the given problem which we call the *incumbent solution*. The subproblems queue contains initially the given problem only. While the subproblems queue is not empty we remove a next subproblem P from this queue and process P in the following way depending on three possible cases.

1. Let P satisfy C0-condition. Then, by Proposition 1, P has no feasible solutions, so we don't make any operations.
2. Let P satisfy C1-condition, so, by Proposition 2, P has the only optimal solution $\tilde{x}$. Then we compare $\tilde{x}$ with the incumbent solution $\tilde{y}$ and, if $f(\tilde{x}) > f(\tilde{y})$, we replace $\tilde{y}$ by $\tilde{x}$.
3. Let P satisfy neither C0-condition nor C1-condition, so P has free variables. Then we decompose P along the free variable with the minimal index and put the obtained decompositions of P into the subproblems queue.

It is easy to see that after termination of solving the given problem by cBnB algorithm the incumbent solution will be an optimal solution of the problem. The *complexity* of solving SSP by cBnB algorithm is the number of subproblems processed during the procedure of solving. Note that, since processed by the algorithm subproblems are decomposed along the free variable with the minimal index, for each processed subproblem P we have $I_P = \{1, 2, \ldots, s\}$ where $0 < s \le n$ and in the third case this subproblem is decomposed along the variable x_{s+1}. Further, we denote the number s by s_P.

The problem resolution process can be demonstrated by a rooted binary tree which we call a *cBnB-tree* of the problem. The subproblems processed by the cBnB algorithm form the set of the cBnB-tree nodes, i.e. the number of the cBnB-tree nodes is equal to the complexity of solving the problem by cBnB algorithm. The root of the cBnB-tree is the given subset sum problem. Each subproblem decomposed by the cBnB algorithm is connected by arcs with the two decompositions of this subproblem. Thus, leaves of the cBnB-tree are subproblems satisfying either C0-condition or C1-condition, and internal nodes of the cBnB-tree are decomposed subproblems. We will mean by C0-leaf (C1-leaf) of the cBnB-tree a leaf satisfying C0-condition (C1-condition). Since each internal node of the cBnB-tree has two childrens the number of the cBnB-tree nodes is equal to $2L - 1$ where L is the number of the cBnB-tree leaves.

Let P be a subproblem processed by the cBnB algorithm. Then by the level of P we will mean the number of distinct ancestors of P in the cBnB-tree, i.e. the given problem has level 0, the decompositions of the given problem have level 1, the decompositions of the decompositions of the given problem have level 2 and so on. In this paper we consider the cBnB algorithm which traverses all subproblems in the cBnB-tree by levels, using BFS search.

4 Frontal scheme of parallelization

We study questions concerning the parallelization of solving SSP by the cBnB algorithm. In particular, we consider the following scheme of parallelization which we call *the frontal scheme*. For applying this scheme it is assumed that potentially unrestricted number of processors can be used. Among used processors we choose one processor which is called *control processor*. All other used processors are called *operable processors*. The frontal scheme realization depends on some positive integer parameter l which we call *the parallelization level* of the scheme. The considered scheme has two stages of computations. At the first stage the control processor solve the given problem by the cBnB algorithm while the subproblems queue contains subproblems of level less than l. Note that after this stage the subproblems queue contains all subproblems which are the cBnB-tree nodes of level l. We will call these subproblems *candidate subproblems*. Then the control processor saves the current incumbent solution of the given problem and sends all candidate subproblems to operable processors (one subproblem to each processor). At the second stage operable processors which receive candidate subproblems solve completely these subproblems by the cBnB algorithm and send the obtained optimal solutions of subproblems back to the control processor that finds optimal solution of the given problem by comparing the received solutions and the saved incumbent solution.

The first stage processing can be presented by the fragment of the cBnB-tree induced by all subproblems processed at the first stage and all candidate subproblems, i.e. all subproblems of level not greater than l in the cBnB-tree. Note that this fragment is a rooted binary tree of depth l. We will call it *the first stage tree* of level l. Note that internal nodes of the first stage tree are subproblems decomposed at the first stage, and leaves of the first stage tree are either the cBnB-tree C0- and C1-leaves of level less than l (which are also called respectively *C0- and C1-leaves* of

the first stage tree) or candidate subproblems (which are called *candidate leaves* of the first stage tree).

By the first stage complexity we mean the number of subproblems processed by the control processor at the first stage, i.e. the total number of internal nodes and C0- and C1-leaves in the first stage tree. By the second stage complexity we mean the maximal complexity of solving by the cBnB algorithm the subproblem sent to an operable processor. The frontal scheme complexity is defined as the sum of the first stage and the second stage complexities (thus in our complexity model we don't take into account the time required for communications between processors and for comparing the received solutions).

Note that the frontal scheme complexity depends on the chosen parallelization level. The problem we address in the paper is what is the optimal value of parallelization level for which the frontal scheme has the minimal complexity. We investigate this problem for the following particular case of subset sum problem:

$$\begin{array}{l} \text{maximize } f(\tilde{x}) = \sum_{i\in N} ax_i, \\ \text{subject to } g(\tilde{x}) = \sum_{i\in N} ax_i \le ka + 1, \\ x_i \in \{0,1\}, i \in N, \end{array} \tag{3}$$

where $a > 1$ and $k \in \{0, 1, 2, \ldots, n-1\}$. This case has been chosen because it is relatively simple and well studied in the literature [26, 27]. It is easy to see (see, e.g.,[28]) that C0-leaves of the cBnB-tree for the problem (3) are subproblems P such that $\sum_{i=1}^{s_P-1} \theta_P(i) = k$ and $\theta_P(s_P) = 1$. These subproblems have one-to-one correspondence with boolean tuples $(\theta_P(1), \theta_P(2), \ldots, \theta_P(s_P), 0, 0, \ldots, 0) \in \{0,1\}^n$ of weight[1] $k+1$. So the cBnB-tree for the problem (3) has $\binom{n}{k+1}$ C0-leaves. It is also easy to see that C1-leaves of the cBnB-tree for the problem (3) are subproblems P such that $\sum_{i=1}^{s_P-1} \theta_P(i) = k - (n - s_p)$ and $\theta_P(s_P) = 0$ which are in one-to-one correspondence with boolean tuples $(\theta_P(1), \theta_P(2), \ldots, \theta_P(s_P), 1, 1, \ldots, 1) \in \{0,1\}^n$ of weight k. So the cBnB-tree for the problem (3) has $\binom{n}{k}$ C1-leaves. Thus in this tree the total number of leaves is $\binom{n}{k+1} + \binom{n}{k} = \binom{n+1}{k+1}$, and the total number of nodes is $2\binom{n+1}{k+1} - 1$. Hence the complexity of solving the problem (3) by the cBnB algorithm is $2\binom{n+1}{k+1} - 1$, i.e. this complexity depends on neither a nor the set of variable indexes. So without loss of generality we denote the problem (3) by $P[n;k]$ and assume that 0-decomposition (1-decomposition) of the problem $P[n;k]$ is the problem $P[n-1;k]$ ($P[n-1;k-1]$).

By $L^{(1)}_{P[n;k]}(l)$ ($L^{(2)}_{P[n;k]}(l)$) we denote the first stage (the second stage) complexity of solving $P[n;k]$ by applying the frontal scheme for the parallelization level l, and by $L_{P[n;k]}(l) = L^{(1)}_{P[n;k]}(l) + L^{(2)}_{P[n;k]}(l)$ we denote the frontal scheme complexity of solving the problem $P[n;k]$. Define $L^{\star}_{P[n;k]} = \min_l L_{P[n;k]}(l)$. A parallelization level $l^{\star}$ is *optimal* for $P[n;k]$ if $L_{P[n;k]}(l^{\star}) = L^{\star}_{P[n;k]}$. Further, we investigate the asymptotic behavior of optimal parallelization levels for problems $P[n;k]$ as n becomes infinite. This problem was considered earlier in [29, 30]. In particular, it is shown in [30] that in the case $n/3 \le k \le 2n/3$ the optimal parallelization level $l^{\star}$ satisfies the relation $l^{\star} = \frac{n}{2} - \frac{1}{4}\log_2 n + O(1)$ and $L^{\star}_{P[n;k]} = \Theta\left(\frac{2^{n/2}}{\sqrt[4]{n}}\right)$. In this work we generalize this result to the case $n/4 < k < 3n/4$. Moreover, we for the first time consider the case of "small" values k. In particular, we compute the optimal parallelization level $l^{\star}$ and the complexity $L^{\star}_{P[n;k]}$ for the case $k \le \frac{3-\sqrt{5}}{4}n$ ($k \ge (1 - \frac{3-\sqrt{5}}{4})n$) and show in this way that the asymptotic behavior of the values $l^{\star}$ and $L^{\star}_{P[n;k]}$ in this case is different from the asymptotic behavior of these values in the case $n/4 < k < 3n/4$. Thus, the main result of this paper is detecting the difference of the asymptotic behavior of the values $l^{\star}$ and $L^{\star}_{P[n;k]}$ for the cases of "big" and "small" values of k.

5 Auxiliary results

We will call two problems $P' = P[n;k']$ and $P'' = P[n;k'']$ *dual problems* if $k' + k'' = n - 1$. Let P', P'' be dual problems which satisfy neither C0-condition nor C1-condition. It is easy to see that 0-decompositions (1-decompositions) of P' satisfy C1-condition (C0-condition) if and only if 1-decompositions (0-decompositions) of P'' satisfy C0-condition (C1-condition). Moreover, if decompositions of P' and P'' satisfy neither C0-condition nor C1-condition then 0-decompositions of P' are dual to 1-decompositions of P'' and 1-decompositions of P' are dual to 0-decompositions of P''. Note also that dual problems P' and P'' have the same complexity of solving by cBnB algorithm. Using these observations, the following fact can be proved.

Proposition 3. *If $k' + k'' = n - 1$ then $L_{P[n;k']}(l) = L_{P[n;k'']}(l)$ for any parallelization level l.*

According to this proposition, without loss of generality we can restrict our consideration to the case of problems $P[n;k]$ for $k \le n/2$. Further, we give an explicit formula for computing $L^{(1)}_{P[n;k]}(l)$ and $L^{(2)}_{P[n;k]}(l)$ in the case when $k \le n/2$ and $l \le n - k$.

Let $k \le n/2$, $l \le n-k$. For computing $L^{(1)}_{P[n;k]}(l)$ denote by T_1 the first stage tree of level l for the problem $P[n;k]$. First

1 By the weight of a boolean tuple we mean the number of 1-components in the tuple.

consider the case $l \le k+1$. Note that in this case all subproblems of $P[n;k]$ processed at the first stage of the frontal scheme satisfy neither C0-condition nor C1-condition, so all these subproblems are decomposed. Thus the tree T_1 is a full binary tree of depth l containing 2^l candidate leaves and 2^l-1 internal nodes, i.e. in this case $L^{(1)}_{P[n;k]}(l) = 2^l - 1$. Now consider the case $k+1 < l \le n-k$. Note that subproblems of $P[n;k]$ processed at the first stage of the frontal scheme can not satisfy C1-condition because of $l \le n-k$, so the tree T_1 can contain only C0-leaves. It is not difficult to see that C0-leaves of T_1 are all subproblems P of $P[n;k]$ such that $k < s_P < l$, $\theta_P(s_P) = 1$ and $\sum_{i=1}^{s_P-1} \theta_P(i) = k$. The number of such subproblems is $\sum_{s=k}^{l-2} \binom{s}{k} = \binom{l-1}{k+1}$. Thus T_1 contains $\binom{l-1}{k+1}$ C0-leaves. Note also that all candidate leaves of T_1 are decompositions of the problem $P[n;k]$ subproblems of level $l-1$ which satisfy neither C0-condition nor C1-condition. Since the problem $P[n;k]$ subproblems of level $l-1$ can not satisfy C1-condition, this is subproblems of level $l-1$ which don't satisfy C0-condition, i.e. subproblems P such that $s_P = l-1$ and $\sum_{i=1}^{s_P} \theta_P(i) \le k$. The number of such subproblems is $\sum_{t=0}^{k} \binom{l-1}{t}$, so T_1 contains $2\sum_{t=0}^{k} \binom{l-1}{t}$ candidate leaves. Thus the total number of leaves in T_1 is $\binom{l-1}{k+1} + 2\sum_{t=0}^{k} \binom{l-1}{t}$, so T_1 contains $\binom{l-1}{k+1} + 2\sum_{t=0}^{k} \binom{l-1}{t} - 1$ internal nodes. Therefore,

$$L^{(1)}_{P[n;k]}(l) = \left[\binom{l-1}{k+1} + 2\sum_{t=0}^{k}\binom{l-1}{t} - 1\right] + \tag{4}$$
$$+ \binom{l-1}{k+1} = 2\sum_{t=0}^{k+1}\binom{l-1}{t} - 1.$$

Thus

$$L^{(1)}_{P[n;k]}(l) = \begin{cases} 2^l - 1 \text{ if } l \le k+1, \\ \\ 2\sum_{t=0}^{k+1}\binom{l-1}{t} - 1 \text{ if } k+1 < l \le n-k. \end{cases} \tag{5}$$

Since $2\sum_{t=0}^{k+1}\binom{l-1}{t} - 1 \le 2^l - 1$, relation (5) implies

$$2\sum_{t=0}^{k+1}\binom{l-1}{t} - 1 \le L^{(1)}_{P[n;k]}(l) \le 2^l - 1. \tag{6}$$

To compute $L^{(2)}_{P[n;k]}(l)$, note that for fixed n the complexity of solving $P[n;k]$ by the cBnB algorithm achieves the maximum value $\binom{n+1}{\lfloor (n+1)/2 \rfloor}$ for $k = \frac{n-1}{2}$ if n is odd and for $k = \frac{n}{2} - 1, \frac{n}{2}$ if n is even. Moreover, for $k < \frac{n}{2}$ this complexity increases as k increases. Using this observations for $k \le n/2$, we conclude that

$$L^{(2)}_{P[n;k]}(l) = \begin{cases} 2\binom{n-l+1}{\lfloor (n-l+1)/2 \rfloor} - 1 \text{ if } l \ge n - 2(k+1), \\ \\ 2\binom{n-l+1}{k+1} - 1 \text{ if } l \le n - 2(k+1). \end{cases} \tag{7}$$

Note also that for $L^{(1)}_{P[n;k]}(l)$ we have the following obvious fact.

Proposition 4. *$L^{(1)}_{P[n;k]}(l') \le L^{(1)}_{P[n;k]}(l'')$ for any l', l'' such that $0 < l' \le l'' \le n$.*

Further, we use the following known lower bound on $\binom{2k}{k}$ (see, e.g., [31]):

$$\binom{2k}{k} \ge \frac{2^{2k}}{2\sqrt{k}}. \tag{8}$$

The following binomial inequality is also used.

Proposition 5. *Let $k \ge 2$, $m \ge 3$, and $q \ge \sqrt[3]{2}m$. Then*

$$\sum_{i=m+1}^{q}\binom{i}{k} \ge \binom{m}{k+1}.$$

Proof. Without loss of generality we assume that $q \le 2m$. Then $q^3 - q \ge 2(m^3 - m)$ which implies

$$\frac{(q+1)q(q-1) - (m+1)m(m-1)}{m(m-1)(m-2)} > 1. \tag{9}$$

Note that

$$\binom{m+1}{k} = \frac{(k+1)(m+1)}{(m-k+1)(m-k)}\binom{m}{k+1} \tag{10}$$
$$\ge \frac{3(m+1)}{(m-1)(m-2)}\binom{m}{k+1}$$
$$= \frac{3(m+1)m}{m(m-1)(m-2)}\binom{m}{k+1}.$$

Moreover, for each $i > m+1$ we have

$$\binom{i}{k} = \frac{i}{i-k}\binom{i-1}{k} \ge \frac{i}{i-2}\binom{i-1}{k}. \tag{11}$$

From (10) and (11) by induction for $i = m+1, m+2, \ldots$ we obtain

$$\binom{i}{k} \ge \frac{3i(i-1)}{m(m-1)(m-2)}\binom{m}{k+1}.$$

Thus

$$\sum_{i=m+1}^{q}\binom{i}{k} \ge \binom{m}{k+1}\sum_{i=m+1}^{q}\frac{3i(i-1)}{m(m-1)(m-2)} \tag{12}$$
$$= \frac{3\binom{m}{k+1}}{m(m-1)(m-2)}\sum_{i=m+1}^{q} i(i-1).$$

It can be easily checked that

$$\sum_{i=m+1}^{q} i(i-1) = \frac{1}{3}[(q+1)q(q-1) - (m+1)m(m-1)],$$

so the proposition follows from (12) and (9).

6 Estimations of the frontal scheme complexity

We consider the two following cases of problems $P[n;k]$: the case $n/2 \geq k \geq \frac{n}{4}(1+\varepsilon)$ where $\varepsilon > 0$ for big values of k and the case $k \leq \frac{3-\sqrt{5}}{4}n$ for small values of k.

6.1 The case for big values of k

Let $n/2 \geq k \geq \frac{n}{4}(1+\varepsilon)$ where $\varepsilon > 0$. Note that for $l \leq n/2$ in this case we have $\sum_{t=0}^{k+1}\binom{l-1}{t} > \frac{1}{2}2^{l-1}$, so it follows from (6) that

$$2^{l-1} \leq L^{(1)}_{P[n;k]}(l) < 2^{l}. \tag{13}$$

Therefore, using Proposition 4, for any $l \geq n/2$ we obtain

$$L_{P[n;k]}(l) \geq L^{(1)}_{P[n;k]}(l) \geq L^{(1)}_{P[n;k]}(\lfloor n/2 \rfloor) \geq 2^{\lfloor n/2 \rfloor - 1}. \tag{14}$$

We consider separately the two following cases.

1. Let $l \leq n-2(k+1)$. Then by relation (7) we have $L^{(2)}_{P[n;k]}(l) = 2\binom{n-l+1}{k+1} - 1$. Thus

$$L_{P[n;k]}(l) = L^{(1)}_{P[n;k]}(l) + 2\binom{n-l+1}{k+1} - 1.$$

For $0 < l \leq n-2(k+1)$ consider $\Delta(l) = L_{P[n;k]}(l-1) - L_{P[n;k]}(l)$. Note that we have

$$\begin{aligned}\Delta(l) &= L^{(1)}_{P[n;k]}(l-1) - L^{(1)}_{P[n;k]}(l) + \\ &\quad + 2\left(\binom{n-l+2}{k+1} - \binom{n-l+1}{k+1}\right) \\ &= L^{(1)}_{P[n;k]}(l-1) - L^{(1)}_{P[n;k]}(l) + 2\binom{n-l+1}{k} \\ &\geq 2\binom{n-l+1}{k} - L^{(1)}_{P[n;k]}(l).\end{aligned}$$

Hence, by relation (13),

$$\Delta(l) > 2\binom{n-l+1}{k} - 2^{l}.$$

Note that if l increases then $2\binom{n-l+1}{k}$ decreases and 2^l increases. So the value $2\binom{n-l+1}{k} - 2^l$ decreases as l increases. Thus the minimal value $2\binom{n-l+1}{k} - 2^l$ is achieved for the maximal value $l = n-2(k+1)$, so for any l

$$\Delta(l) > 2\binom{n-(n-2(k+1))+1}{k} - 2^{n-2(k+1)}$$

$$= 2\binom{2k+3}{k} - 2^{n-2k-2} > 2\binom{2k}{k} - 2^{n-2k-2}.$$

Let n be sufficiently large such that $2^{n\varepsilon} > \sqrt{n}/4$. Using inequality (8), we obtain

$$\begin{aligned}\Delta(l) &> 2\frac{2^{2k}}{2\sqrt{k}} - 2^{n-2k-2} = \frac{2^{2k}}{\sqrt{k}} - 2^{n-2k-2} \\ &> \frac{2^{2k}}{\sqrt{n}} - 2^{n-2k-2} = 2^{n-2k-2}\left(\frac{2^{4k}}{2^{n-2}\sqrt{n}} - 1\right) \\ &\geq 2^{n-2k-2}\left(\frac{2^{4\frac{n}{4}(1+\varepsilon)}}{2^{n-2}\sqrt{n}} - 1\right) \\ &= 2^{n-2k-2}\left(\frac{2^{n\varepsilon}}{\sqrt{n}/4} - 1\right)\end{aligned}$$

Therefore, from $2^{n\varepsilon} > \sqrt{n}/4$ we derive $\Delta(l) > 0$, i.e. $L_{P[n;k]}(l)$ is a monotonically decreasing function for $0 \leq l \leq n-2(k+1)$. Thus, in this case, for sufficiently large values of n, $L_{P[n;k]}(l)$ has the only minimum value for $l = n-2(k+1)$, i.e. any optimal parallelization level for $P[n;k]$ is not less than $n-2(k+1)$.

2. Now let $l \geq n-2(k+1)$. Then by relation (7), using well-known asymptotics $\binom{n}{\lfloor n/2 \rfloor} \sim \frac{2^n}{\sqrt{\pi n/2}}$, we have

$$\begin{aligned}L^{(2)}_{P[n;k]}(l) &\sim 2\binom{n-l+1}{\lfloor (n-l+1)/2 \rfloor} \\ &\sim \frac{2^{n-l+2}}{\sqrt{\pi(n-l+1)/2}} \sim \frac{4}{\sqrt{\pi/2}} \cdot \frac{2^{n-l}}{\sqrt{n-l}}.\end{aligned} \tag{15}$$

Therefore, for sufficiently large values of n,

$$3\frac{2^{n-l}}{\sqrt{n-l}} < L^{(2)}_{P[n;k]}(l) < 4\frac{2^{n-l}}{\sqrt{n-l}}. \tag{16}$$

Denote $l_0 = \lfloor \frac{n}{2} - \frac{1}{4}\log_2 n \rfloor$. Note that, for sufficiently large values of n such that $\varepsilon n \geq \frac{1}{2}\log_2 n$, we have $n-2(k+1) \leq l_0 \leq n/2$, so, using inequalities (13) and (16), we obtain

$$\begin{aligned}L_{P[n;k]}(l_0) &< 2^{l_0} + 4\frac{2^{n-l_0}}{\sqrt{n-l_0}} \\ &\leq 2^{\frac{n}{2}-\frac{1}{4}\log_2 n} + 4\frac{2^{\frac{n}{2}+\frac{1}{4}\log_2 n+1}}{\sqrt{n/2}} \\ &< 13\frac{2^{n/2}}{\sqrt[4]{n}}.\end{aligned}$$

Thus $L^{\star}_{P[n;k]} < 13\frac{2^{n/2}}{\sqrt[4]{n}}$.
Consider the three following subcases.

a) Let $l \leq l_0 - 3$. Then, by inequalities (16),

$$\begin{aligned}L_{P[n;k]}(l) \geq L^{(2)}_{P[n;k]}(l) &> 3\frac{2^{n-l}}{\sqrt{n-l}} \\ &> 3\frac{2^{\frac{n}{2}+\frac{1}{4}\log_2 n+3}}{\sqrt{n}} = 24\frac{2^{\frac{n}{2}}}{\sqrt[4]{n}} > L^{\star}_{P[n;k]}.\end{aligned}$$

b) Let $n/2 \geq l \geq l_0 + 6$. Then, by inequalities (13),

$$L_{P[n;k]}(l) \geq L^{(1)}_{P[n;k]}(l) \geq 2^{l-1} \geq 2^{l_0+5} > 2^{\frac{n}{2}-\frac{1}{4}\log_2 n+4} = 16\frac{2^{n/2}}{\sqrt[4]{n}} > L^{\star}_{P[n;k]}.$$

c) Let $l \geq n/2$. Then, by inequality (14), $L_{P[n;k]}(l) \geq 2^{\lfloor n/2 \rfloor - 1}$, so, for sufficiently large values of n, we have $L_{P[n;k]}(l) \geq 13\frac{2^{n/2}}{\sqrt[4]{n}} > L^{\star}_{P[n;k]}$.

Thus, for sufficiently large values of n, any l satisfying subcases a)–c) cannot be optimal for $P[n;k]$, so any optimal parallelization level $l^{\star}$ for $P[n;k]$ satisfies the inequalities $l_0 - 2 \leq l^{\star} \leq l_0 + 5$, i.e. $l^{\star} = \frac{n}{2} - \frac{1}{4}\log_2 n + O(1)$. Therefore, using relations (13) and (15), we can easily obtain

$$L^{\star}_{P[n;k]} = L_{P[n;k]}(l^{\star}) = \Theta\left(\frac{2^{n/2}}{\sqrt[4]{n}}\right).$$

6.2 The case for small values of k

Let $0 < k \leq \frac{3-\sqrt{5}}{4}n$. Note that in this case $l \leq n - 2(k+1)$, so by relation (7) we have $L^{(2)}_{P[n;k]}(l) = 2\binom{n-l+1}{k+1} - 1$. Further, we assume that $n \geq 5(\sqrt{5}+2)$, i.e. $n \geq 22$ (the case $n < 22$ can be checked immediately). Then the inequality $k \leq \frac{3-\sqrt{5}}{4}n$ implies that $n \geq 4k + 5$. We consider the four following cases.

1. Let $l \leq k + 2$. Then from relation (5) we can derive $L^{(1)}_{P[n;k]}(l) = 2^l - 1$. Thus

$$L_{P[n;k]}(l) = 2^l + 2\binom{n-l+1}{k+1} - 2.$$

Therefore, if we denote again $\Delta(l) = L_{P[n;k]}(l-1) - L_{P[n;k]}(l)$,

$$\begin{aligned}\Delta(l) &= 2\left(\binom{n-l+2}{k+1} - \binom{n-l+1}{k+1}\right) - 2^{l-1} \\ &= 2\binom{n-l+1}{k} - 2^{l-1} \\ &\geq 2\binom{n-l+1}{k} - 2^{k+1}.\end{aligned}$$

Note that

$$\binom{n-l+1}{k} = \frac{(n-l+1)!}{k!(n-l+1-k)!} = \prod_{i=1}^{k}\frac{n-l+2-i}{k+1-i} \geq \left(\frac{n-l+1}{k}\right)^k \geq \left(\frac{n-k-1}{k}\right)^k.$$

Since $k \leq \frac{3-\sqrt{5}}{4}n < n/4$, we have $\frac{n-k-1}{k} \geq \frac{n-2k}{k} > 2$, so $\binom{n-l+1}{k} > 2^k$. Therefore, $\Delta(l) > 0$, i.e. $L_{P[n;k]}(l)$ is a monotonically decreasing function for $0 \leq l \leq k + 2$. Thus, in this case $L_{P[n;k]}(l)$ has the only minimum value for $l = k + 2$, i.e. any optimal parallelization level for $P[n;k]$ is not less than $k + 2$.

2. Let $k + 2 \leq l \leq n - 2(k+1)$. Then by relation (5) we have $L^{(1)}_{P[n;k]}(l) = 2\sum_{i=0}^{k+1}\binom{l-1}{i} - 1$. Thus

$$L_{P[n;k]}(l) = 2\sum_{i=0}^{k+1}\binom{l-1}{i} + 2\binom{n-l+1}{k+1} - 2. \quad (17)$$

For convenience consider separately the case $k = 1$. In this case

$$L_{P[n;k]}(l) = 2l^2 - 2(n+1)l + n^2 + n,$$

so $L_{P[n;k]}(l)$ has the minimum value $L_{P[n;k]}(\frac{n+1}{2})$ if n is odd and the minimum values $L_{P[n;k]}(\frac{n}{2})$ and $L_{P[n;k]}(\frac{n}{2}+1)$ if n is even, i.e. for any n the minimum value of $L_{P[n;k]}(l)$ is achieved at $l = \lceil n/2 \rceil$. Now let $k > 1$. Denote $\delta(l) = L_{P[n;k]}(l+1) - L_{P[n;k]}(l)$. Then for $k + 2 \leq l < n - 2(k+1)$

$$\begin{aligned}\delta(l) &= \left(2\sum_{i=0}^{k+1}\binom{l}{i} + 2\binom{n-l}{k+1} - 2\right) - \\ &- \left(2\sum_{i=0}^{k+1}\binom{l-1}{i} + 2\binom{n-l+1}{k+1} - 2\right) \\ &= 2\left(\sum_{i=0}^{k+1}\left(\binom{l}{i} - \binom{l-1}{i}\right) - \right. \\ &- \left.\left(\binom{n-l+1}{k+1} - \binom{n-l}{k+1}\right)\right) \\ &= 2\left(\sum_{i=1}^{k+1}\binom{l-1}{i-1} - \binom{n-l}{k}\right) \\ &= 2\left(\sum_{i=0}^{k}\binom{l-1}{i} - \binom{n-l}{k}\right).\end{aligned}$$

Note that if l increases then the sum $\sum_{i=0}^{k}\binom{l-1}{i}$ increases and $\binom{n-l}{k}$ decreases, so $\delta(l)$ increases as l increases, i.e. for any $k + 2 \leq l' < l'' < n - 2(k+1)$

$$\delta(l') < \delta(l''). \quad (18)$$

Further, we consider separately the cases for odd and even values of n.

a) Let n be odd, i.e. $n = 2n' + 1$. Then we have

$$\delta(n'+1) = 2\left(\sum_{i=0}^{k}\binom{n'}{i} - \binom{n'}{k}\right)$$

$$= 2\sum_{i=0}^{k-1}\binom{n'}{i} > 0.$$

Now we prove that $\delta(n') < 0$. Note that

$$\delta(n') = 2\left(\sum_{i=0}^{k}\binom{n'-1}{i} - \binom{n'+1}{k}\right)$$

where

$$\begin{aligned}\binom{n'+1}{k} &= \binom{n'}{k} + \binom{n'}{k-1}\\ &= \binom{n'-1}{k} + 2\binom{n'-1}{k-1}\\ &\quad + \binom{n'-1}{k-2}.\end{aligned}$$

Thus

$$\begin{aligned}\delta(n') &= \sum_{i=0}^{k}\binom{n'-1}{i} - \binom{n'+1}{k}\\ &= \sum_{i=0}^{k-3}\binom{n'-1}{i} - \binom{n'-1}{k-1}\\ &= \sum_{i=0}^{k'-2}\binom{n'-1}{i} - \binom{n'-1}{k'},\end{aligned}$$

where $k' = k-1$. So the inequality $\delta(n') < 0$ is obvious for $k \le 3$. Let $k > 3$. Note that

$$\frac{\binom{n'-1}{i-1}}{\binom{n'-1}{i}} = \frac{i}{n'-i}.$$

So for $0 < i \le k'-2$ we have

$$\binom{n'-1}{i-1} \le \frac{k'-2}{n'-k'+2}\binom{n'-1}{i}.$$

Therefore

$$\begin{aligned}\sum_{i=0}^{k'-2}\binom{n'-1}{i} &< \binom{n'-1}{k'-2}\left(1 + \frac{k'-2}{n'-k'+2} + \right.\\ &\quad \left. + \left(\frac{k'-2}{n'-k'+2}\right)^2 + \dots\right)\\ &= \binom{n'-1}{k'-2}\frac{1}{1-\frac{k'-2}{n'-k'+2}}\\ &= \binom{n'-1}{k'-2}\frac{n'-k'+2}{n'-2k'+4}\\ &< \binom{n'-1}{k'-2}\frac{n'-k'+2}{n'-2k'}.\end{aligned}$$

On the other hand,

$$\binom{n'-1}{k'} = \frac{(n'-k'+1)(n'-k')}{k'(k'-1)}\binom{n'-1}{k'-2}.$$

Moreover,

$$\begin{aligned}\frac{n'-k'+1}{k'-1} &= 1 + \frac{n'-2k'+2}{k'-1}\\ &> 1 + \frac{n'-2k'+2}{k'} = \frac{n'-k'+2}{k'},\end{aligned}$$

so

$$\binom{n'-1}{k'} > \frac{(n'-k'+2)(n'-k')}{k'^2}\binom{n'-1}{k'-2}.$$

Thus

$$\begin{aligned}&\sum_{i=0}^{k'-2}\binom{n'-1}{i} - \binom{n'-1}{k'}\\ &< \binom{n'-1}{k'-2}\frac{n'-k'+2}{n'-2k'} -\\ &\quad -\binom{n'-1}{k'-2}\frac{(n'-k'+2)(n'-k')}{k'^2}\\ &= \binom{n'-1}{k'-2}(n'-k'+2)\left(\frac{1}{n'-2k'} - \frac{n'-k'}{k'^2}\right).\end{aligned}$$

It is easy to check that for $k' \le \frac{3-\sqrt{5}}{2}n'$ the inequality $\frac{1}{n'-2k'} - \frac{n'-k'}{k'^2} \le 0$ is valid, hence

$$\sum_{i=0}^{k'-2}\binom{n'-1}{i} - \binom{n'-1}{k'} < 0.$$

Thus $\delta(n') < 0$ for $k' \le \frac{3-\sqrt{5}}{2}n'$ (which follows from $k \le \frac{3-\sqrt{5}}{4}n$). Taking into account inequalities (18), from $\delta(n'+1) > 0$ and $\delta(n') < 0$ we conclude that the minimum value of $L_{P[n;k]}(l)$ for $k+2 \le l \le n-2(k+1)$ is achieved at $l = n'+1$.

b) Now let n be even, i.e. $n = 2n'$. For $l = n'$ we have

$$\begin{aligned}\delta(n') &= 2\left(\sum_{i=0}^{k}\binom{n'-1}{i} - \binom{n'}{k}\right)\\ &= 2\left(\sum_{i=0}^{k}\binom{n'-1}{i} - \binom{n'-1}{k}\right.\\ &\quad \left. -\binom{n'-1}{k-1}\right)\\ &= 2\sum_{i=0}^{k-2}\binom{n'-1}{i} > 0.\end{aligned}$$

For $l = n' - 1$ we have

$$\delta(n'-1) = 2\left(\sum_{i=0}^{k}\binom{n'-2}{i} - \binom{n'+1}{k}\right) < 2\left(\sum_{i=0}^{k}\binom{n'-2}{i} - \binom{n'}{k}\right).$$

Thus, by the same way as in the case of odd n we can prove that for $k' \le \frac{3-\sqrt{5}}{2}(n'-1)$ (which follows from $k \le \frac{3-\sqrt{5}}{4}n$)

$$\sum_{i=0}^{k}\binom{n'-2}{i} - \binom{n'}{k} < 0,$$

so $\delta(n'-1) < 0$. From $\delta(n') > 0$, $\delta(n'-1) < 0$ and inequalities (18) we can conclude that the minimum value of $L_{P[n;k]}(l)$ is achieved at $l = n'$.

Summing up the cases a) and b), we obtain that for $k+2 \le l \le n-2(k+1)$ the minimum value of $L_{P[n;k]}(l)$ is achieved at $l = \lceil n/2 \rceil$.

3. Let $n-2(k+1) < l \le n-k$. Note that for $n-2(k+1) \le l \le n-k$

$$L_{P[n;k]}(l) = 2\sum_{i=0}^{k+1}\binom{l-1}{i} + 2\binom{n-l+1}{\lfloor (n-l+1)/2 \rfloor} - 2.$$

So for $n-2(k+1) \le l < n-k$

$$\delta(l) = 2\left[\left(\sum_{i=0}^{k+1}\binom{l}{i} - \sum_{i=0}^{k+1}\binom{l-1}{i}\right) - \tau(l)\right] = 2\left[\sum_{i=0}^{k}\binom{l-1}{i} - \tau(l)\right]$$

where $\tau(l) = \binom{n-l+1}{\lfloor (n-l+1)/2 \rfloor} - \binom{n-l}{\lfloor (n-l)/2 \rfloor}$. Using Pascal's rule, it is easy to see that

$$\tau(l) = \begin{cases} \binom{n-l}{\lfloor (n-l)/2 \rfloor - 1} \text{ if } n-l \text{ is even,} \\ \binom{n-l}{\lfloor (n-l)/2 \rfloor + 1} \text{ if } n-l \text{ is odd.} \end{cases} \quad (19)$$

It can be easily checked from (19) that τ_l decreases monotonically as l increases, i.e. $\tau(l'') < \tau(l')$ for $n-2(k+1) \le l' < l'' < n-k$. Thus, since $\sum_{i=0}^{k}\binom{l-1}{i}$ increases monotonically as l increases, we obtain that $\delta(l)$ also increases as l increases, i.e. $\delta(l) \ge \delta(n-2(k+1))$ for $n-2(k+1) \le l < n-k$. Moreover, taking into account that $\tau(n-2k-2) = \binom{2k+2}{k}$ by (19) and $n \ge 4k+5$, we have

$$\tau(n-2(k+1)) = 2\left[\sum_{i=0}^{k}\binom{n-2k-3}{i} - \binom{2k+2}{k}\right] > 2\left[\binom{n-2k-3}{k} - \binom{2k+2}{k}\right] \ge 0,$$

i.e. $\tau(n-2(k+1)) > 0$. Thus, $\delta(l) > 0$ for $n-2(k+1) \le l < n-k$, so in this case $L_{P[n;k]}(l) > L_{P[n;k]}(n-2(k+1))$, i.e. l can not be an optimal parallelization level for $P[n;k]$.

4. Let $l > n-k$. Then, using Proposition 4, we have

$$L_{P[n;k]}(l) > L^{(1)}_{P[n;k]}(l) \ge L^{(1)}_{P[n;k]}(n-k) = 2\sum_{i=0}^{k+1}\binom{n-k-1}{i} - 1. \quad (20)$$

On the other hand, taking into account $k+2 \le \lceil n/2 \rceil \le n-2(k+1)$, from (17) we obtain that

$$\begin{aligned} &L_{P[n;k]}(\lceil n/2 \rceil) \quad (21)\\ &= 2\sum_{i=0}^{k+1}\binom{\lceil n/2 \rceil - 1}{i} + 2\binom{n-\lceil n/2 \rceil + 1}{k+1} - 2 \\ &\le 2\sum_{i=0}^{k+1}\binom{\lfloor n/2 \rfloor}{i} + 2\binom{\lfloor n/2 \rfloor + 1}{k+1} - 2. \end{aligned}$$

Further, we prove the inequality

$$\sum_{i=0}^{k+1}\binom{\lfloor n/2 \rfloor}{i} + \binom{\lfloor n/2 \rfloor + 1}{k+1} < \sum_{i=0}^{k+1}\binom{n-k-1}{i} \quad (22)$$

For $k = 1$ this inequality is checked directly. Let $k \ge 2$. Denote $n' = \lfloor n/2 \rfloor \ge 11$. Note that $n \ge 4k+5 > 3k+6$, so $n-k-2 > 2n/3 \ge 4n'/3 > \sqrt[3]{2}n'$. Thus, using Proposition 5, we have

$$\sum_{i=n'+1}^{n-k-2}\binom{i}{k} \ge \binom{n'}{k+1}.$$

Then, taking into account the equality

$$\binom{n-k-1}{k+1} = \binom{n'+1}{k+1} + \sum_{i=n'+1}^{n-k-2}\binom{i}{k}$$

obtained by sequential applying of Pascal's rule, we derive

$$\binom{n-k-1}{k+1} \ge \binom{n'+1}{k+1} + \binom{n'}{k+1}. \quad (23)$$

From $n-k-1 > n'$ we have also

$$\sum_{i=0}^{k}\binom{n-k-1}{i} > \sum_{i=0}^{k}\binom{n'}{i}.$$

Summing up this inequality with (23), we obtain inequality (22) for $k \geq 2$. It follows from inequalities (20), (21) and (22) that $L_{P[n;k]}(l) > L_{P[n;k]}(\lceil n/2 \rceil)$, i.e. in this case l can not also be an optimal parallelization level for $P[n;k]$.

Summarizing all the considered cases for small values of k, we can conclude that for big enough n the parallelization level $\lceil n/2 \rceil$ is optimal for $P[n;k]$, and, moreover, $\lceil n/2 \rceil$ is only optimal parallelization level for $P[n;k]$ if $k \geq 2$ or n is odd (if $k = 1$ and n is even the parallelization level $\lceil n/2 \rceil + 1$ is also optimal for $P[n;k]$).

7 Conclusion

In conclusion we formulate the obtained results.

Theorem 1. *Let* $n/2 \geq k \geq \frac{n}{4}(1+\varepsilon)$ *for some fixed* $\varepsilon > 0$, *and* $l^\star$ *be an optimal parallelization level for* $P[n;k]$. *Then* $l^\star = \frac{n}{2} - \frac{1}{4}\log_2 n + O(1)$ *and* $L^\star_{P[n;k]} = \Theta\left(\frac{2^{n/2}}{\sqrt[4]{n}}\right)$.

Theorem 2. *Let* $0 < k \leq \frac{3-\sqrt{5}}{4}n$. *Then the value* $l^\star = \lceil n/2 \rceil$ *is an optimal parallelization level for* $P[n;k]$, *and* $L^\star_{P[n;k]} = \Theta\left(\binom{\lfloor n/2 \rfloor + 1}{k+1}\right)$.

It follows from the obtained results that in the cases for big and small values of k the optimal parallelization level $l^\star$ for $P[n;k]$ and the value $L^\star_{P[n;k]}$ have distinct asymptotic behaviors. We conjecture that in the case of $\frac{3-\sqrt{5}}{4}n < k \leq \frac{n}{4}(1-\varepsilon)$ for some fixed $\varepsilon > 0$ the relation $l^\star = n/2 + O(1)$ holds, i.e. the value $k = n/4$ is the boundary for the considered cases of asymptotic behavior of the values $l^\star$ and $L^\star_{P[n;k]}$.

Note also that our results are obtained under the condition that potentially unrestricted number of processors can be used while in a practical situation the number of processors is limited. In the case when the number of available processors is not enough for the optimal parallelization, according to our results, it can be concluded that for the most efficient solving of the problem one has to use as much as possible processors.

Acknowledgement: This work is partially supported by Russian Foundation for Fundamental Research (Grant 18-07-00566).

References

[1] Kellerer H., Pfershy U., Pisinger D., Knapsack Problems, Springer Verlag, 2004

[2] Posypkin M., Sin S. T. T., Comparative analysis of the efficiency of various dynamic programming algorithms for the knapsack problem, Proceedings of 2016 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference (EIConRusNW), 2016, 313–316

[3] Posypkin M., Sigal I., Speedup estimates for some variants of the parallel implementations of the branch-and-bound method, Computational Mathematics and Mathematical Physics, 2006, 46, N 12, 2187–2202

[4] Afanasiev A., Bychkov I., Zaikin O., Manzyuk M., Posypkin M., Semenov A., Concept of a multitask grid system with a flexible allocation of idle computational resources of supercomputers, Journal of Computer and Systems Sciences International, 2017, 56, N 4, 701–707

[5] Bellman R.E., Dynamic Programming, Princeton University Press, Princeton, 1957

[6] Horowitz E., Sahni S., Computing partitions with applications to the knapsack problem, Journal of ACM, 1974, 21, N 2, 277–292

[7] Kindervater G.A.P., Lenstra J.K., An introduction to parallelism in combinatorial optimization, Discrete Applied Mathematics, 1986, 14, 135–156

[8] Lee J., Shragowitz E., Sahni S., A hypercube algorithm for the 0/1 knapsack problem, Journal of Parallel and Distributed Computing, 1988, 5, 438–456

[9] Lin J., Storer J., Processor efficient hypercube algorithm for the knapsack problem, Journal of Parallel and Distributed Computing, 1991, 3, 332–337

[10] Sanches C.A.A., Somaa N.Y., Yanasse H.H., Parallel time and space upper-bounds for the subset-sum problem, Theoretical Computer Science, 2008, 407, 342-348

[11] Curtis V.V., Sanches C.A.A., An efficient solution to the subset sum problem on GPU. Concurrency Computation: Practice and Experience, 2016, 28, 95–113

[12] Sanches C.A.A., Soma N.Y., Yanasse H.H., An optimal and scalable parallelization of the two-list algorithm for the subset-sum problem, European Journal of Operational Research, 2007, 176, 870–879

[13] Bokhari S.S., Parallel solution of the subset-sum problem: an empirical study, Concurrency and Computation: Practice and Experience, 2012, 24, 2241–2254

[14] Curtis V.V., Sanches C.A.A., A low-space algorithm for the subset-sum problem on GPU, Computers & Operations Research, 2017, 83, 120–124

[15] Kang L., Wan L., Li K., Efficient Parallelization of a Two-List Algorithm for the Subset-Sum Problem on a Hybrid CPU/GPU Cluster, Proceedings of International Symposium on Parallel Architectures, Algorithms and Programming, 2014, 93–98

[16] Wan L., Li K., Liu J., Li K., GPU implementation of a parallel two-list algorithm for the subset-sum problem, Concurrency and Computation: Practice and Experience, 2015, 27, 119–145

[17] Wan L., Li K., Li K., A novel cooperative accelerated parallel two-list algorithm for solving the subset-sum problem on a hybrid CPU/GPU cluster, Journal of Parallel and Distributed Computing, 2016, 97, 112–123

[18] Ristovski Z., Mishkovski I., Gramatikov S., Filiposka S., Parallel implementation of the modified subset sum problem in CUDA, Proceedings of 22nd Telecommunications Forum Telfor (TELFOR), Belgrade, 2014, 923–926

[19] Curtis V.V., Sanches C.A.A., An improved balanced algorithm for the subset-sum problem, European Journal of Operational Research, 2019, 275, 460–466

[20] Barkalov K., and Gergel V., Parallel global optimization on GPU, Journal of Global Optimization, 2016, 66, N 1, 3–20

[21] Pietracaprina A., Pucci G., Silvestri F., Vandin F., Space-efficient parallel algorithms for combinatorial search problems, Journal of Parallel and Distributed Computing, 2015, 76, 58–65

[22] Casado L. G., Martinez J. A., García I., Hendrix E. M. T., Branch-and-bound interval global optimization on shared memory multiprocessors, Optimization Methods & Software, 2008, 23, N 5, 689–701

[23] Vu T.-T., Derbel B., Parallel Branch-and-Bound in multi-core multi-CPU multi-GPU heterogeneous environments, Future Generation Computer Systems, 2016, 56, 95–109

[24] Baldwin A., Asaithambi A., An efficient method for parallel interval global optimization, Proceedings of 2011 International Conference on High Performance Computing and Simulation (HPCS), 2011, 317–321

[25] Kolpakov R., Posypkin P., The lower bound on complexity of parallel branch-and-bound algorithm for subset sum problem, AIP Conference Proceedings, 2016, 1776, N 1, AIP Publishing

[26] Finkel'shtein Yu., Priblizhennye metody i prikladnye zadachi diskretnogo programmirovaniya, Nauka, Moscow, 1976 (in Russian)

[27] Grishukhin V., Efficiency of the branch and bound method in problems with boolean variables, Issledovaniya po diskretnoj optimizatsii, Nauka, Moscow, 1976, 203–230 (in Russian)

[28] Kolpakov R., Posypkin M., Upper and lower bounds for the complexity of the branch and bound method for the knapsack problem, Discrete Mathematics and Applications, 2010, 20, N 1, 113–125

[29] Kolpakov R., Posypkin M., Sigal I., On a lower bound on the computational complexity of a parallel implementation of the branch-and-bound method, Automation and Remote Control, 2010, 71, N 10, 2152–2161

[30] Kolpakov R., Posypkin M., Estimating the computational complexity of one variant of parallel realization of the branch and bound method for the knapsack problem, J. of Computer and Systems Sciences International, 2011, 50, N 5, 756–765

[31] Koshy T., Catalan numbers with applications, Oxford University Press, USA, 2009