Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2014) 22: 1382 – 1394

doi:10.3906/elk-1210-105

Performance of exhaustive search with parallel agents

Toni DRAGANOV STOJANOVSKI*
University of Information Science and Technology "St. Paul the Apostle", Ohrid, Macedonia

Received: 24.10.2012 • **Accepted:** 15.02.2013 • **Published Online:** 15.08.2014 • **Printed:** 12.09.2014

Abstract: The advent of high-performance computing via many-core processors and distributed processing emphasizes the possibility for exhaustive search by multiple search agents. Despite the occurrence of elegant algorithms for solving complex problems, exhaustive search has retained its significance since many real-life problems exhibit no regular structure and exhaustive search is the only possible solution. Here we analyze the performance of exhaustive search when it is conducted by multiple search agents. Several strategies for joint search with parallel agents are evaluated. We discover that the performance of the search improves with the increase in the level of mutual help between agents. The same search performance can be achieved with homogeneous and heterogeneous search agents provided that the lengths of subregions allocated to individual search regions follow the differences in the speeds of heterogeneous search agents. We also demonstrate how to achieve the optimum search performance by means of increasing the dimensions of the search region.

Key words: Parallel algorithms, exhaustive search, multiagent systems

1. Introduction

Exhaustive search consists of systematically enumerating all possible candidates for the solution and checking whether each candidate satisfies the problem's statement. The number of candidate solutions to consider grows very rapidly with problem size, causing lengthy or even infeasible searches.

However, the continuing increase in computing power and memory sizes, and the advent of many-core processors and parallel and distributed programming [1, 2], increases the feasibility of exhaustive search and has revived interest in brute-force techniques for a good reason. An overview of parallel search algorithms for solving discrete optimization problems is given in [3]. Many real-life problems reveal no regular structures to be exploited in the search for solutions, and this leaves exhaustive search as the only possible approach. We can always try to design elegant and optimal algorithms in a quest for order of magnitude of performance improvements. However, mathematical creativity is not guaranteed to give success for real-life problems. As pointed out many times in the past, it is more likely that order of magnitude improvements can be achieved due to program optimization and the clever use of computational resources in an exhaustive search.

Though considered inelegant, exhaustive search proved to be feasible for several problems. The SETI@home project is searching for extraterrestrial intelligence in narrow-bandwidth radio signals from space using a virtual supercomputer composed of large numbers of Internet-connected computers [4]. It was launched in May 1999; as of 2008, 5 million people in 226 countries have volunteered their PCs to analyze data. Combined, their PCs form the world's second most powerful supercomputer, averaging 482 TeraFLOPs and contributing over 2

*Correspondence: toni.stojanovski@uist.edu.mk

million years of CPU time [5]. None of the proposed cryptanalytic attacks against DES [6] since its adoption as an encryption standard in 1976 were practically feasible. Eventually DES was broken in 1998 using exhaustive search of its 56-bit key space by a custom-made machine named "Deep Crack" built by the Electronic Frontier Foundation [7].

Advances in computing technology in the second half of the twentieth century made exhaustive search feasible and applicable in other areas, too. The 4-color theorem was the first theorem to be proven using a computer [8]. Exhaustive search for large Mersenne prime numbers has been going on continuously since 1996 [9]. A more detailed analysis of the application of exhaustive search can be found in [10, 11]. We dare to expect that other complex real-life problems will soon become "victims" of exhaustive search due to increasing numbers of processing cores in many-core CPUs, advances in distributed computing, or other technological breakthroughs.

Exhaustive search needs not necessarily to test all candidate solutions. A subregion of the entire region may be skipped (search tree is pruned) if a failed candidate implies that the subregion cannot contain a solution. Backtracking, constraint propagation, and consistency checking [12] are efficient techniques for pruning the search tree. Our results are also applicable to other search strategies provided that the search region can be divided into disjoint regions, i.e. each agent first searches its own region before continuing the search in the regions allocated to neighboring agents.

In this paper we evaluate the performance of exhaustive search when it is conducted with many search agents working in parallel. Dependence of performance of exhaustive search with parallel agents on the following parameters is analyzed:

- Differences in speeds of search agents,
- Length of allocated search subregions,
- Type of communication between central server and agents.

We will also determine the optimum division of the search region into subregions, i.e. the division that minimizes the average search time.

According to the taxonomy defined in [13], if search agents implement the same algorithm with the same parameter configuration, then the resulting search is called *homogeneous*. Homogeneous search agents perform the search with the same speed. When the search agents implement different algorithms or the same algorithm but with distinct configurations, then the strategy is called *heterogeneous*. The speed of search for heterogeneous agents is different. If the agents communicate during the search, such strategies are called cooperative multiagent strategies. Otherwise, they are called independent search strategies. In this paper, we consider independent search agents: they do no communicate during the search, but they conduct the search in parallel. A consequence of considering independent search agents is that our results for heterogeneous agents are equally valid for agents implementing the same algorithm with different speeds and for agents implementing different algorithms and with different speeds. In [13, 14], the cooperative strategy for homogeneous and heterogeneous agents is analyzed. Adding cooperation to homogeneous agents speeds up the search. The authors in [13, 14] also observed an advantage in using heterogeneous versus homogeneous cooperation. In [15] we presented results on the performance of exhaustive search. The results were obtained mostly through numerical simulations. Here 3 propositions and 1 corollary are added to mathematically prove the results from [15].

Here is the overview of the paper: in Section 2, we present theoretical results on the average search time for exhaustive search. Section 3 analyzes exhaustive search with homogeneous parallel agents, while Section 4 analyzes joint search with heterogeneous agents. Section 5 concludes the paper and gives directions for future research.

2. Exhaustive search methods

First we give a formal definition of exhaustive search. Consider function $F : X \to [0, 1]$, where X is the discrete finite N-dimensional domain of F. X is also called the search region. Assume that there is only one point $x_s \in X$ such that $F(x_s) = 1$. Otherwise, $F(x) = 0$ if $x \in X$ and $x \neq x_s$. Point x_s is called the solution of function F. We assume uniform probability mass function (pmf) of the solution x_s in the search region X; that is, each point in the search region X is equally probable to be the solution x_s.

If function F exhibits no regular structure, then finding the solution x_s by exhaustively searching the domain X can be the only option. Exhaustive search attempts to find the point x_s by repeatedly calculating $F(x)$ for all points $x \in X$ until x_s is found such that $F(x_s) = 1$.

We analyze the case where the region is jointly searched by m agents $a_1, a_2, \ldots, a_m$. We are interested in the impact of the joint search with multiple search agents on the performance of the search. Search performance is measured by the average time required to find the only solution x_s.

For sake of simplicity and clarity, we analyze exhaustive search of *one-dimensional* region $X = [1, L]$. However, results are equally valid for *multidimensional* regions since they can be easily converted into one-dimensional regions. A uniformly increasing set of boundary values $b_i \in X, i = 1, 2, \ldots, m$ divides X into m disjoint subregions $X_1, X_2, \ldots, X_m$. Thus, $\bigcup_{i=1}^{m} X_i = X$. The relation between boundary points and subregions is given by

$$X_i = [b_i, b_{(i+1) \mod m}). \tag{1}$$

Each agent a_i is allocated a subregion X_i, which it searches with speed v_i. Unless otherwise stated, in the rest of the paper one-directional search is employed: each agent starts its search from the starting point of the allocated subregion. This is illustrated in Figure 1.

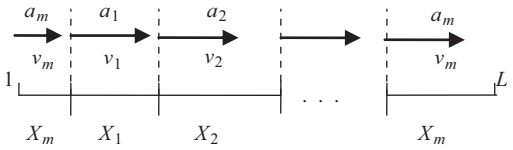

Figure 1. Exhaustive search with multiple search agents.

A summary of the notation used in this paper is given next:

$|X|$ Length of region X;

$p_s(l)$ Probability that the solution x_s is in a region with length l, i.e. $\Pr\{x_s \in X_i || X_i| = l\}$;

$p_l(l)$ pmf of length l of subregions;

$p_v(v)$ pmf of searching speed v of search agents;

$p_{v,l}(v,l)$ Joint pmf of v and l, i.e. the probability that a subregion with length l is searched by an agent with speed v.

The process of allocation of search subregions to search agents depends on (i) the type of communication between search agents and the central server, (ii) whether the central server knows the number m of search agents and the speed of each agent, and (iii) whether the number of search agents is determined before the search starts or additional agents can join the search at a later time. One possibility is that the central server can allocate a subregion to each search agent depending on the number m of search agents and the speed of each agent. Another possibility is that each agent chooses its own starting point for the search irrespective of the starting points and the speeds of the other search agents. The search continues until a solution is found, or a "stop" command is received from the central server. The following proposition addresses the performance of exhaustive search as measured by the average search time.

Proposition 1 *Consider a division of a one-dimensional search region $X = [1, L]$ into m subregions with random length l that are jointly searched by m agents $a_1, a_2, \ldots, a_m$. Searching speeds v of search agents are independent identically distributed (iid) random variables. If l and v are mutually independent random variables, then the average search time is given by*

$$E(t) = \frac{m}{2L} E(l^2) E(\frac{1}{v}). \tag{2}$$

Proof Average search time is calculated by averaging the search time over the agent's speed v and region's length l using the following formula:

$$E(t) = \sum_{l} \sum_{v} p_v(v) p_s(l) \frac{l}{2v}, \tag{3}$$

where $l/(2v)$ is the average time to find the solution in a region of length l by an agent with speed v. One can easily show that

$$p_s(l) = \frac{m p_l(l) l}{L}, \tag{4}$$

where $mp_l(l)$ is the number of regions with length l, and $mp_l(l)l$ is the total number of points belonging to regions with length l. Then the ratio $mp_l(l)l/L$ is the probability that the solution x_s is in a region with length l. Substituting Eq. (4) into Eq. (3), one obtains

$$E(t) = \frac{m}{2L} \sum_{l} \sum_{v} p_v(v) p_l(l) \frac{l^2}{v} = \frac{m}{2L} E(\frac{1}{v}) E(l^2). \tag{5}$$

□

Next we analyze the case where the speed of the agent allocated to a subregion depends on the subregion's length. The following corollary stems from Proposition 1.

Corollary 1 *If v and l are mutually dependent random variables with joint pmf $p_{v,l}(v,l)$, then the average search time is given by*

$$E(t) = \frac{m}{2L} E(\frac{l^2}{v}). \tag{6}$$

Proof Similar to Eq. (3), average search time is calculated as

$$E(t) = \sum_{l} \sum_{v} p_{s,v}(l,v) \frac{l}{2v}, \tag{7}$$

where $p_{s,v}(l,v) = \Pr\{v, x_s \in X_i || X_i| = l\}$ denotes the probability that the solution x_s is in region X_i with length l and that the region is searched by an agent with speed v. Then, similar to Eq. (4), joint pmf for v and l is given by

$$p_{v,l}(v,l) = \frac{mlp_{s,v}(l,v)}{L}. \tag{8}$$

Substituting Eq. (8) into Eq. (7), one obtains

$$E(t) = \frac{m}{2L} \sum_{l} \sum_{v} p_{v,l}(v,l) \frac{l^2}{v}. \tag{9}$$

□

3. Exhaustive search with homogeneous agents

In this section we additionally assume that all search agents are homogeneous with the same speed V; that is,

$$p_v(v) = \begin{cases} 1 & \text{for } v = V \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Next, 3 methods for division of the search region are explained and compared.

3.1. Equal subregions

First we consider the case where 2-directional communication exists between the central server and search agents, and the number of search agents m is known in advance. As shown later on in Proposition 3, the central server can minimize the average search time if it divides the search region into m equal subregions and then assigns each subregion to a different search agent. In this case,

$$p_l(l) = \begin{cases} 1 & \text{for } l = \frac{L}{m} \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

Thus, Eq. (2) yields

$$E(t) = \frac{m}{2L} \frac{1}{V} \frac{L^2}{m^2} = \frac{L}{2mV}. \tag{12}$$

3.2. Semiequal subregions

Next we analyze the case where the number of search agents is not known in advance and 2-directional communication exists between central server and search agents. New search agents can register at run time. The search subregion is allocated and communicated to each search agent by the central server as it registers.

The search subregion allocated to a newly registered agent depends on the current number of search agents. The first agent will start its search from point 1. If there is only one search agent in the system, then it searches the whole region. If a second search agent joins the search, then its subregion is the second half of the whole region; that is, the second agent will start the search from point $1+L/2$. The third agent, when it registers, will start from point $1+L/4$, and the fourth agent will start from point $1+3L/4$. The next agents will start the search from points $1+L/8$, $1+3L/8$, $1+5L/8$, $1+7L/8$, etc. Thus, the search subregions are shrinking as the number of search agents grows. Each time a new search agent joins the search, it is given to search the second half of the currently largest search subregion. In the *semiequal subregions* method, for $m = 2^0, 2^1, 2^2, 2^3, \ldots$, $p_l(l)$ and $E(t)$ are again given by Eqs. (11) and (12), respectively, thus giving the same average time as for the *equal subregions* method. Otherwise, for $2^i < m < 2^{i+1}$, the probability that a search subregion is with length l is given by

$$p_l(l) = \begin{cases} \dfrac{2^{i+1}-m}{m} & \text{for } l = \dfrac{L}{2^i} \\ \dfrac{2m-2^{i+1}}{m} & \text{for } l = \dfrac{L}{2^{i+1}} \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

The average search time calculated using Eq. (13) is higher than the average search time for the *equal subregions* method from Eq. (12).

3.3. Random subregions

Finally, we consider a case where the number of search agents is not known in advance and one-directional communication exists from search agents to central server. One-directional communication is used by a search agent to communicate to the central server when the solution is found. Each search agent starts from a randomly chosen starting point, thus randomly choosing its search subregion. Therefore, the size of the subregion searched by an agent can vary between 1 and the size L of the entire search region. The following Proposition gives the pmf for length l of search subregions.

Proposition 2 *Consider a one-dimensional region $X = [1, L]$ that is divided into m disjoint subregions by a set of boundary points $b_i, i = 1, 2, \ldots, m$. Boundary points are iid random variables with uniform distribution in X. Then the length of the subregions is a random variable $l \in \{1, 2, \ldots, L\}$ with pmf*

$$p_l(l) = \begin{cases} \dfrac{1}{L^{m-1}} & \text{for } l = L \\ \left(\dfrac{L-l}{L}\right)^{m-1} \displaystyle\sum_{i=1}^{m-1} \binom{m-1}{i} (L-l)^{-i} & \text{otherwise.} \end{cases} \tag{14}$$

Proof We analyze the length of subregion X_1 starting from b_1 and determine its pmf. The same discussion applies to all subregions. Without loss of generality, we assume that $b_1 = 1$. We consider 2 cases: $l = L$ and $l \neq L$.

X_1 is with length $l = L$ if all $m-1$ boundary points $b_2, b_3, \ldots, b_m$ are equal to 1 and thus coincide with b_1. The probability that a boundary point $b_i = 1$ for $i = 2, 3, \ldots, m$ is $1/L$. Thus, $p_l(L) = 1/L^{m-1}$, which is identical to the first part of Eq. (14).

X_1 is with length $l \neq L$ if $1 \leq i \leq m-1$ boundary points are equal to $l+1$, while the other $m-1-i$ boundary points take values from the set $\{1, l+2, l+3, \ldots, L\}$. The probability that a boundary point takes

a value from the set $\{1, l+2, l+3, \ldots, L\}$ is $\frac{L-l}{L}$. Then for $l \neq L$, it follows that

$$p_l(l) = \sum_{i=1}^{m-1} \binom{m-1}{i} (\frac{L-l}{L})^{m-1-i} (\frac{1}{L})^i, \tag{15}$$

which is identical to the second part of Eq. (14). □

Figure 2 depicts the pmf $p_l(l)$ for $L = 1000$ and $m = 2, 3, 5, 10, 20$ calculated using Eq. (14). For $m = 2$, $p_l(l)$ is a uniformly distributed pmf in the region $[1, L]$, while for $m = 3$, $p_l(l)$ is a linearly decreasing pmf in the region $[1, L]$.

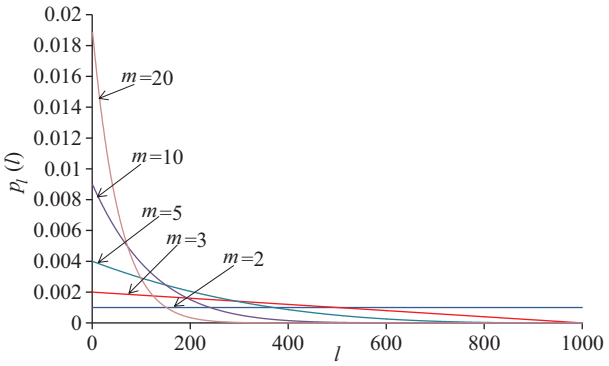

Figure 2. Probability mass function $p_l(l)$ for $L = 1000$ and $m = 2, 3, 5, 10, 20$ calculated using Eq. (14).

Figure 3 shows that smaller l become more probable for larger m. For $m = 10$, 1% of the smallest lengths, i.e. $l = 1, \ldots, 1000$, contains 8.65% of the probability mass, while for $m = 256$ the same set of lengths contains 92.29% of the probability mass.

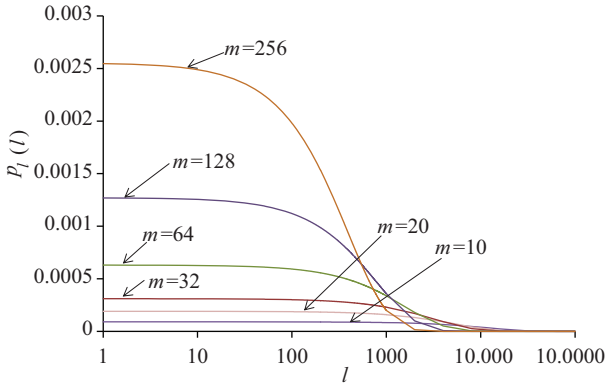

Figure 3. Probability mass function $p_l(l)$ for $L = 100,000$ and $m = 10, 20, 32, 64, 128, 256$ calculated using Eq. (14). Note that logarithmic scale is used for the x-axis to demonstrate that for larger m the pmf becomes more concentrated around smaller l.

3.4. Comparison

Figure 4 compares the average search time (y-axis) for the 3 methods depending on the number m of homogeneous search agents (x-axis). All search agents are with speed $V = 1$. Average search time is calculated substituting Eqs. (11), (13), and (14) for $p_l(l)$ into Eq. (2). Region length is $L = 100000$. L does not affect the shape of the 3 curves. A different value for L will only change the scale for the x-axis.

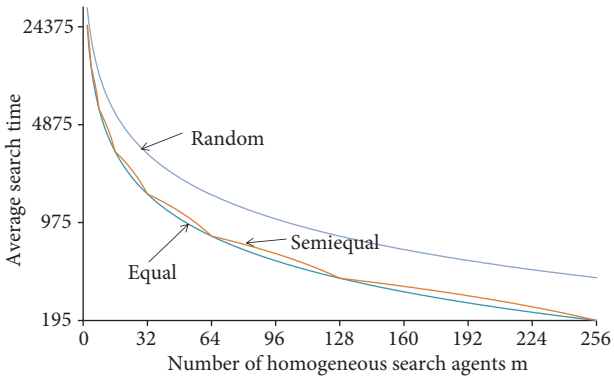

Figure 4. Comparison of average search times of the 3 searching methods for $L = 100,000$.

As expected, the *equal subregions* method produces the best performance, i.e. the shortest average search time. The *semiequal subregions* method produces performances that are close to the ones produced by the *equal subregions* method. The *random subregions* method results in significantly higher average search times. For example, 10 search agents using the *equal subregions* method will produce the same performance as 19 search agents using the *random subregions* method, and 16 search agents using the *equal subregions* method will produce the same performance as 31 search agents using the *random subregions* method.

4. Exhaustive search with heterogeneous agents

In this section we consider the performance of exhaustive search when the parallel agents are heterogeneous. Average search time is calculated using Eq. (5) and Eq. (6). We analyze the performance of exhaustive search with parallel heterogeneous agents for the following 3 search strategies.

4.1. Strategy 1: Subregion's length is proportional to agent's speed

Similar to Section 3.1, we consider the case where 2-directional communication exists between the central server and search agents, and the number of search agents m is known in advance. On basis of the knowledge of the speeds $v_1, v_2, \ldots, v_m$ of the search agents, the central server can decide on the length of each subregion allocated to each agent. A question naturally arises: what is the optimum division of the search region into subregions that minimizes the average search time? The following proposition provides the answer.

Proposition 3 *Consider a one-dimensional search region $X = [1, L]$ that is jointly searched by m agents $a_1, a_2, \ldots, a_m$ with searching speeds $v_1, v_2, \ldots, v_m$. Then the minimum value for the average search time is*

$$E(t) = \frac{L}{2\sum_{i=1}^{m} v_i} \tag{16}$$

and it is achieved if each agent a_i is allocated a subregion X_i with length

$$l_i = \frac{v_i L}{v_1 + v_2 + \ldots + v_m}. \tag{17}$$

The proof follows straightforwardly from applying the first derivative test [16] to find the minimum value for the average search time, which is defined as

$$E(t) = \sum_{i=1}^{m} \frac{l_i}{L} \frac{l_i}{2v_i} = \frac{1}{2L} \sum_{i=1}^{m} \frac{l_i^2}{v_i}. \tag{18}$$

Thus, in the optimum strategy for division of the search region for heterogeneous agents, each search agent is allocated a search subregion whose length is proportional to the agent's speed. Faster agents get larger subregions and slower agents get smaller subregions, thus achieving static load balancing.

For homogeneous agents, average search time is minimized when all agents are allocated subregions with equal length (see Section 3.1). Heterogeneous agents can achieve the same minimum value for the average search time as homogeneous agents if the average speed for the heterogeneous agents $\frac{1}{m}\sum_{i=1}^{m} v_i$ is equal to the speed V of homogeneous agents (see Eqs. (12) and (16)). Curve 'e' in Figure 5 gives the average search time (y-axis) for the optimum strategy depending on the number of heterogeneous search agents (x-axis) when $\frac{1}{m}\sum_{i=1}^{m} v_i = 1$. Its values are identical to the values of curve 'Equal' from Figure 4.

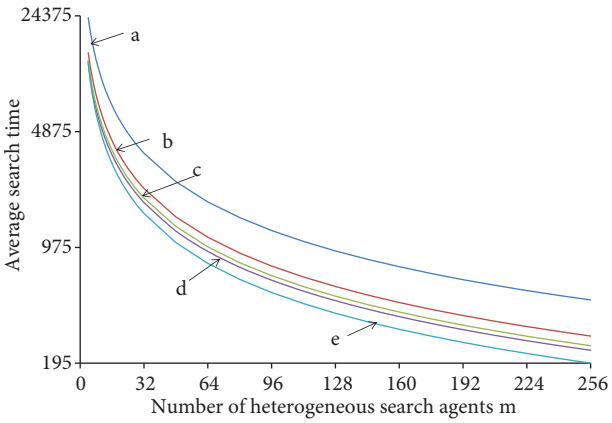

Figure 5. Average search time for parallel heterogeneous search agents for $L = 100,000$ and $p_v(v)$ given by Eq. (20). a- One-directional search; b- joint search in groups of 2 agents; c- joint search in groups of 3 agents; d- joint search in groups of 4 agents; e- length of subregion is determined by agent's speed.

A question naturally arises: if the numbers of agents and their speeds are not known to the server, and there is only one-directional communication from agents to server, then can we still achieve the minimum value for the average search time as in Eq. (16)? We answer this question in the following subsections.

4.2. Strategy 2: One-directional search

Each agent randomly chooses its starting point for the search. The starting point is a uniformly distributed random variable in the region $[1, L]$. Length of allocated search subregions is a random variable whose pmf is

given by Eq. (14) and is depicted in Figure 2. Then each agent is searching its own subregion (see Figure 1). It is possible that agent a_i finishes the search of its region X_i before the solution x_s is found by any of the agents. Then agent a_i continues with the search of the subregion X_{i+1} until a "stop" command is received from the central server. Let v_{min} and v_{max} denote the minimum and maximum speed of search agents, and let l_{min} and l_{max} denote the minimum and maximum length of search subregions. Then, provided that

$$\frac{v_{max}}{v_{min}} < \frac{l_{min} + l_{max}}{l_{max}}, \tag{19}$$

agent a_{i+1} will finish searching subregion X_{i+1} before agent a_i will finish searching both regions X_i and X_{i+1} for all $i = 1, 2, \ldots, m$. In other words, each agent is responsible for the search of its chosen subregion and will receive no help from other agents in the search of its subregion.

As an illustration, we have analyzed heterogeneous agents with speed that is a random variable with the following pmf:

$$p_v(v) = \begin{cases} 0.3 & \text{for } v = 0.5 \\ 0.3 & \text{for } v = 1.0 \\ 0.4 & \text{for } v = 1.375. \end{cases} \tag{20}$$

Note that $E(v) = 1$ and $E(1/v) = 1.191$ for the pmf from Eq. (20). Curve 'a' in Figure 5 gives the average search time (y-axis) for random subregions and one-directional search depending on the number of heterogeneous search agents (x-axis), and it is calculated using Eqs. (2) and (14). Since $E(1/v) = 1.191$ for the pmf from Eq. (20), it follows from Eq. (14) that the average search time for random subregions and heterogeneous agents (curve 'a' from Figure 5) is 1.191 times higher than the average search time for random subregions and homogeneous agents with $V = 1$ (curve 'Random' from Figure 4). This is despite the fact that the average speed of heterogeneous agents from Eq. (20) is equal to the speed V of homogeneous agents.

4.3. Strategy 3: Two-directional search

Similar to the previous case, each agent randomly chooses a starting point. However, neighboring agents can help each other in the following manner: each agent a_i conducts the search in 2 directions, to the left and to the right of the chosen starting point (see Figure 6). If agent a_i's speed is v_i, then the search to the left side is conducted with speed $v_i/2$. The same speed applies for the search to the right side. Searching in both

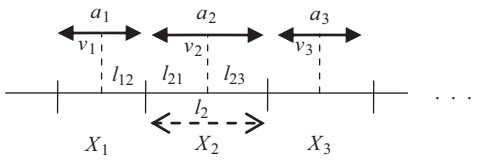

Figure 6. Exhaustive search with parallel agents in 2 directions.

directions improves the assistance between neighboring agents. Assume that the distance between the starting points for agents a_1 and a_2 is d_{12}, and the distance between the starting points for agents a_2 and a_3 is d_{23}. If a_2 is faster than a_1, then a_2 will search a larger portion of d_{12} than a_1. Similarly, if a_2 is slower than a_3, then a_2 will search a smaller portion of d_{23} than a_3. In other words, agents help their slower neighbors

and get help from faster neighbors. It is rather straightforward to show that, for example, agent a_2 searches a subregion with length $l_2 = l_{21} + l_{23}$ where $l_{21} = \frac{v_2 d_{12}}{v_2 + v_1}$ and $l_{23} = \frac{v_2 d_{23}}{v_2 + v_3}$. Time spent by agent a_2 to search the subregion X_2 is given by $t_2 = (l_{21} + l_{23})/v_2$.

Curve 'b' in Figure 5 depicts the average search time for random subregions and 2-directional search for $p_v(v)$ given in Eq. (20). It is obvious that the search strategy with 2 neighbors helping each other (2-directional search) significantly reduces the search time compared to the one-directional search strategy where the neighboring agents do not help each other. An obvious question arises: can the search time be further reduced if agents help each other in groups of 3, i.e. if agents in groups of 3 jointly search an allocated subregion? Curve 'c' in Figure 5 confirms our intuitive expectations: joint search of subregions with groups of 3 agents further reduces the average search time. Further increasing the number of agents that jointly search a subregion to 4 additionally reduces the search time, as shown in curve 'd' from Figure 5. The validity of these conclusions is not dependent on the particular $p_v(v)$ chosen in Eq. (20); that is, a lower average search time is achieved for a higher number of agents jointly searching a subregion irrespective of the pmf $p_v(v)$. Curves 'b', 'c', and 'd' are calculated numerically using Monte Carlo simulations [17].

Defining the strategy for joint search in groups of 3 and more neighbors is beyond the scope of this paper. Here we are interested only in the search performance. Still, Figure 7 gives a possible strategy for joint search by groups of 4 search agents in a 2-dimensional search region X. Each agent randomly chooses its starting point from X and then searches in 4 directions simultaneously, i.e. in 2 directions for each of the 2 dimensions. If the speed of agent a_i is v_i, then the search speed in each of the 4 directions will be $v_i/4$. Subregion X_{22} searched by agent a_{22} is shown as a shaded square in Figure 7. Careful examination of the boundaries of the subregion X_{22} reveals that $v_{22} < v_{23}, v_{22} > v_{32}, v_{22} > v_{12}, v_{22} = v_{21}$.

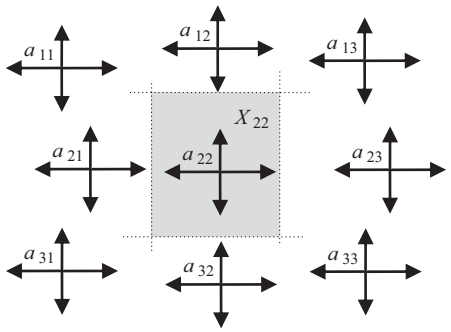

Figure 7. Exhaustive search in a 2-dimensional search region and mutual assistance with 4 neighbors.

Figure 7 also gives a hint of a possible strategy for joint search with $n = 2^N$ neighboring agents for $N > 2$. If the search region is N-dimensional, then each agent chooses a random starting point and performs the search in $2N$ directions, 2 directions for each dimension. If the dimension of the search region X is less than N, then one first needs to rearrange the L points from X onto a N-dimensional lattice. Then agents randomly choose starting points from the newly created N-dimensional search region.

4.4. Comparison

As we see from Figure 5, in the case of heterogeneous agents, the average search time decreases and the search performance improves as the mutual assistance between agents grows. One-directional search is the worst

search strategy: mutual assistance is reduced to the division of the search region between the search agents. If faster agents are enabled to help slower agents, e.g., by searching the allocated one-dimensional subregion in 2 directions, then the average search time reduces dramatically. If the number of agents n that jointly search a subregion grows, then the average search time is further reduced. For example, one-directional search with $m = 256$ agents, 2-directional search with $m = 155$ and $n = 2$, 2-directional search with $m = 136$ and $n = 3$, 2-directional search with $m = 127$ and $n = 4$, and optimum search with $m = 107$ produce similar average search times. For $n = m$, the 2-directional search strategy converges to the optimum strategy and each agent searches a subregion whose size is proportional to the agent's speed.

5. Conclusion

We have analyzed the performance of exhaustive search with parallel search agents. Both homogeneous and heterogeneous agents were analyzed. Performance of exhaustive search with parallel search agents improves and average search time decreases as the level of mutual assistance increases. Optimum performance is achieved if the central server knows the number and speeds of search agents and there is 2-directional communication between central server and agents. Then each agent is allocated a search subregion with length proportional to agent's speed. Even if the number and speeds of search agents are not known and there is one-directional communication from agents to the central server, search performances close to the optimum can still be achieved: the search region needs to have high-dimension N close to the number of agents m and 2-directional search needs to be employed.

Several questions still remain open. What is the impact of the presence of multiple solutions x_s on the average search time when we want to find one solution or all solutions? What is the impact of the tree pruning on the performance of exhaustive search? The amount of tree pruning is not known at the time of subregion allocation, which can result in load imbalancing [12]. Dynamic load balancing can be exploited in this case: if an agent finishes its search prior to the other agents due to significant tree pruning, then another subregion needs to be additionally allocated to this agent. Can the 2-directional search in a high-dimensional search region as explained in Section 4.3 be generalized to other search methods such as backtracking, constraint propagation, or consistency checking with the aim of reducing the agent's idle periods and agent–central server communication?

References

[1] G.R. Andrews, Foundations of Multithreaded, Parallel, and Distributed Programming, Boston, MA, USA, Addison-Wesley, 1999.

[2] J. Sanders, E. Kandrot, CUDA by Example: An Introduction to General-Purpose GPU programming, Boston, MA, USA, Addison-Wesley, 2010.

[3] A. Grama, V. Kumar, "State of the art in parallel search techniques for discrete optimization problems", IEEE Transactions on Knowledge and Data Engineering, Vol. 11, pp. 28–35, 1999.

[4] E. Korpela, D. Werthimer, D. Anderson, J. Cobb, M. Lebofsky, "SETI@home–massively distributed computing for SETI", Computing in Science and Engineering, Vol. 3, pp. 78–83, 2011.

[5] A. Siemion, J.V. Korff, P. McMahon, E. Korpela, D. Werthimer, D. Anderson, G. Bower, J. Cobb, G. Foster, M. Lebofsky, J. van Leeuwen, M. Wagner, "New SETI sky surveys for radio pulses", Acta Astronautica: Special Issue on Searching for Life Signatures, Vol. 67, pp. 1342–1349, 2010.

[6] W. Stallings, Cryptography and Network Security: Principles and Practice, Upper Saddle River, NJ, USA, Prentice Hall, 2005.

[7] P.C. Cocher, "Breaking DES", CryptoBytes, Vol. 4, pp. 1–5, 1999.

[8] K. Appel, W. Haken, "The solution of the four-color-map problem", Scientific American, Vol. 237, pp. 108–121, 1977.

[9] Mersenne Research, Inc."Great Internet Mersenne Prime Search", online, available at http://www.mersenne.org/.

[10] J. Nievergelt, "Exhaustive search, combinatorial optimization and enumeration: exploring the potential of raw computing power", in SOFSEM 2000 LNCS 1963, pp. 18–35, 2000.

[11] J. Nievergelt, R. Gasser, F. Mser, C. Wirth, "All the needles in a haystack: can exhaustive search overcome combinatorial chaos?", Lecture Notes in Computer Science, Vol. 1000, pp. 254–274, 1995.

[12] V. Kumar, "Algorithms for constraint satisfaction problems: a survey", AI Magazine, Vol. 13, pp. 32–44, 1992.

[13] A. Masegosa, D. Pelta, I. del Amo, J. Verdegay, "On the performance of homogeneous and heterogeneous cooperative search strategies", in Nature Inspired Cooperative Strategies for Optimization (NICSO 2008), Vol. 236, pp. 287–300, 2009.

[14] A. Masegosa, PhD, "Cooperative methods in optimisation: analysis and results", University of Granada, Granada, Spain, 2010.

[15] T. Stojanovski, L. Krstevski, "On the performance of exhaustive search with cooperating agents", ETAI Conference, Ohrid, Macedonia, 2011.

[16] R. Smith, R. Minton, Calculus: Multivariable, New York, NY, USA, McGraw-Hill Education, 2001.

[17] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, "Equation of state calculations by fast computing machines", Journal of Chemical Physics, Vol. 21, pp. 1087–1092, 1953.