Objective Analysis in Task Planning and Allocation of Multicomputer Systems

A. Velarde M.

Instituto Tecnológico El Llano, Aguascalientes,
Mexico

Abstract. Parallel computing systems with multiple processing elements have the ability to run a set of different tasks at the same time. To achieve such a parallelism, these systems use typically an algorithm for task planning and another algorithm for task assignment. The planner algorithm must solve the problem of how many and which tasks remaining in a queue must be executed, how many processors should be used to execute the selected tasks, and which tasks must be executed first; meanwhile, the assignment algorithm must determine what free processors in a mesh will be used to execute the selected tasks. The objectives of both algorithms are maximizing the use of all processors and the adjacency of the processors assigned to a same task, as well as minimizing the waiting times of the tasks in a queue. Nonetheless, in the purpose of maximizing and minimizing all the objectives at the same time, several conflicts may occur among them, causing degradation in the performance of a parallel computing system. In this paper, it is presented an analysis of how the objectives in the task planning and assignment may conflict, specifically in multicomputer systems. The analysis is carried out by using a multi-objective optimization algorithm, through which each objective is evaluated to determine its effect in the performance of a parallel computing system. With the results of the evaluated objectives, a scale of priorities is proposed.

Keywords: Task planning, parallel computing

1 Introduction

Multicomputer systems with architectural meshes, interconnection topologies in 2D and 3D, denominated multicomputers in 2D or 3D mesh for commercial purposes and researching, have been the most common parallel systems due to their simplicity, scalability, structural regularity and simple implementation [1,17,3] in research and industry environments.

Various parallel computers commercial and experimental, such as the IBM BlueGene/L [4] and the Intel Paragon [5] have been built based on these two architectures. Some of the commercial multicomputer systems are Multiple Instruction Multiple Data (MIMD), with architectures that permit processor sub mesh partitions, and have the advantage of supporting multiple processes. Each of which can be assigned to an independent processor sub mesh for execution.

In an MIMD mesh, that supports multiple users, a task must be assigned to a free processor submesh, that corresponds to the size required of the operating system. The tasks solicit different computing requirements, and processor submeshes with different sizes within the mesh. When a task is finished executing, the submesh that it occupied is freed up for the next assignment process, this is known as consecutive assignment. The task assignment problem in multicomputer systems can be approached on two levels: on a task level and on a programming level [1,8]. For this research the task level assignment is used.

The main problem with efficient utilization of the processors in dynamic mesh multiuser systems, is the planning of computing resources [6,7,8]. Mesh resource planning, through hardware partitioning involves two components: task planning and a task assignment to the mesh. The function of task assignment, is to choose the next task or tasks for the queue that will be assigned to a free sub mesh to be executed. The function of submesh assignment, is to localize free submeshes that are to be assigned to the selected tasks for planning [6]. In figure 1, where the busy processors are shown with dark circles and the free processors in white, we can see a joining of 6 tasks in the queue to be ingresses into a 2D mesh with a processor size 8×8.

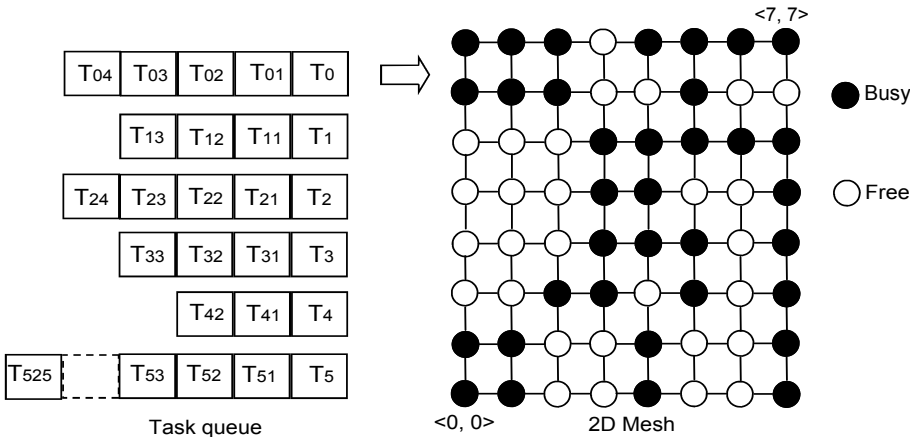

Fig. 1. System structure for task execution in a multi-computer 2D mesh system.

In task assignment for mesh processors there are two different methods: the continuous assignment method, in which the assignments are carried out only in adjacent processors within the mesh, and the non continuous assignment method which allows tasks to be assigned to processors, that are not found to be adjacent to the mesh. In the carrying out of planning and assignment functions,

independent from the type of assignment that is to be used, the following 6 criteria are aimed to be minimized or maximized [9]: system utilization, processor performance, average stationary time, wait time, remain time, result coefficient and the performance coefficient. These objectives, upon being evaluated with task loads in the system, generally result in counter-position, due to the fact that bettering one result in turn worsens another. Thus provoking the parallel system into being highly efficient under one criteria, and in others having efficient results far under the established norms [11,12].

In [13], five of these six objectives are evaluated through an Evolutionary Distribution Algorithm (EDA), especially being the case with the Unified Marginal Distribution Algorithm (UMDA). The object evaluation in [13], is meant to obtain the assignment which is best adjusted, to certain threshold values that are established as optimum values in the planning and assignment tasks. However, with the individualized evaluation process, the results are contrasted producing a multi-objective problem.

An example of the aforementioned is presented when the adjacency between processors assigned to tasks is maximized, thus making that the time in which the tasks remain, and wait in the queue becomes maximized, encouraging degradation in response times that the parallel system authorizes to the users.

In [10], a multi-objective problem is defined as, that in which involves the optimizing of a number of objectives simultaneously. With these types of problems, the objectives are in conflict with each other, the optimal solution of each function that corresponds with each objective (function objective) is different from the rest. In solving these problems, with or without the presence of constraints, it results in a set of interchangeable optimal solutions, popularly known as Pareto optimal solutions [10]. Due to the multiplicity in solutions, these problems were proposed to be solved appropriately using Evolutionary Optimization Algorithms (EOA), those in which use a population focus in the search engine procedure. Evolutionary Optimization Algorithms, use a population based approach, in which more than one solution is involved in an iteration, and evolve a new population of solutions at each iteration [10]. Multi-objective optimization problems give rise to a set of solutions, which require further processing to achieve a single main solution. To perform the first task, a natural proposition is to use an EOA, because the use of a population in one iteration helps an EOA to simultaneously find multiple non-dominated solutions, representing an exchange between objectives in a single simulation run. So considering that planning and allocating tasks in a parallel system, is a multi-objective problem, in this paper, five of the conflicting objectives in the planning and allocation of tasks on a system multicomputers are raised (defined below).

The objectives that the algorithm for scheduling must cover during implementation and proposed in [13] are:

1. Reducing the remain time of the tasks in the queue.
2. Decrease task starvation ,which would avoid discrimination in the allocation of tasks that require a lot of processors (great tasks). This is caused because

the tasks that require a small amount of processors (small jobs), are being continuously assigned.

The task allocation algorithm during its execution, is responsible for covering the following objectives proposed in [13]:

1. Reduce the number of assignments to the mesh of processors performing the tasks allocation algorithm.
2. Maximizing the use of the mesh processors, i.e., decrease the percentage of processors that remain free after the allocation algorithm places, one or more tasks in the mesh of processors (external fragmentation) [11].
3. Maximize contiguity between processors (assign the set of free allocate processors as close together as possible), to minimize the distance in the communication path, and avoid interference between them [12]; this is done in order to get a good parallel algorithm to decrease communication time, and maximize processing time [12].

The selection of the five objectives presented above obeys the completion of a state of the art review, of the research work related to different techniques or proposed planning and task allocation methods, seeking, isolated, to optimize one or more of the six criteria or objectives for planning and tasking.

In this paper, we address the problem of planning and allocation of tasks to a mesh of processors as a multiobjective problem, using for this the evolutionary algorithm outlined in [13] and explained in this section: UMDA algorithms (from this research), evaluating each objective function for analysis of the results, and determine which are the determinants or is the better decision to assign, and schedule them in a system of multicomputers objectives.

For the completion of the objective contrast analysis, a multicomputer Liebres InTELigentes system for teaching programming, systems of high performance computing in higher education institutions, [14] was used to perform the analysis of the contrast between the objectives. The results of each of the objective functions evaluated, obtained with different workloads in the queue of the target system, and with processors mesh sizes 4×4, 8×8 and 16×16. When the values of the objective functions are similar, a priority scale is established for the proposed set of objectives, in this way, a successive application of this scale takes places until the best allocation is found, as set forth in [13].

This research is divided as follows: a section about previous works, where the research conducted in the area of the allocation of processors is presented, a section of basic concepts where the terms related to a 2D architecture mesh are defined; a section titled UMDA algorithm where performance and features of this algorithm are proposed, as well as it is presented in [13] and how it adapts to this research this algorithm. Processes for the analysis of the contrast between the objectives, the planning and allocation of tasks in a 2D mesh and contrast of the objectives during the planning and allocation of tasks, is explained by a set of examples and how the objectives are opposed, during the planning and allocation of tasks on a system multicomputers. In the section of experiments,

tests for the analysis of conflicting goals and resolve the opposition that occurs in the goals set during the planning, and allocation of tasks in the screen are explained. In the last section, the conclusions are drawn from the analysis and experiments conducted are presented.

2 Previous work

Several investigations have been conducted to develop strategies for assignments, in both contiguous and non-contiguous parallel computing systems. This section describes, for reasons of space, only the most significant non-contiguous allocation techniques, that have been developed within different researches. It has been possible to extract the characteristics of the methods, and define the scale priorities of the different objectives proposed in this paper, which seek to become maximized or minimized.

The first adjustment technique FF (First Fit) [15], through the pursuit of free submesh, and determining the sub-mesh that best fits the application, seeks to find the maximum adjacency between processors to lower latency communication between tasks. In the paging technique [15], the iterative process of division of the sub-mesh in partitions of equal size $2i$, where i is a positive integer representing the index of the page parameter, seeks to assign a task to a selected page, allowing the job to run with a total of processors avoiding interference adjacency messages by disjoint processors. MBS [15], a process of division of the mesh to obtain overlapped square submeshes with potency lengths of 2, recursively will decrease to be suited to a request, this causes the work to be embedded on a set of 100% adjacent processors. In ANCA [16], it first tries to assign the task to a sub-mesh of adjoining processor, if it fails, the application is partitioned into sub-partitions of equal size recursively, until it is able to assign subpartitions in locations available to the mesh. In the Random strategy [15], tasks are assigned to the mesh depending on a random number and all free processors are considered in the allocation, with this type of arbitrary allocation use of all available processors and the elimination of any kind of fragmentation that can occur are sought, however, a high communication interference occurs between tasks.

Newer techniques have similar connotations to the original proposals, through the use of an initial strategy to assign the tasks, but when the allocation is not able to be done, a second strategy that replaces the first to achieve the objective of allocation is activated. Examples of such techniques, include search strategy and friendly Multiple Adaptive (Adaptive Scan and Multiple Buddy AS & MB) [15], Allocation Contiguous No Quick (QNA for its acronym in English Quick Non-Contiguous Allocation) [18], and strategy allocation proposed in [17]. In AS & MBS, it seeks to assign the task to a sub-grid of equal size as the one requested, if it does not exist, the MBS strategy is activated to perform the division process of requirements [19].

The strategy proposed in citeBani, the FF method is used in conjunction with the BF method, as follows: if a task requests a sub-mesh sized 4×4 and

the application can not be granted, the request size is reduced to a multiple of 2, for this case 2×2 mesh size requested, and so on until the request has the minimum number of processors, in this case 1×1. When the first technical fault occurs, a second technique is the BF is activated through this technique, a search is performed in free submeshes that best fit it, i.e., with the exact number of processors that the task requires [20]. In fact, 2 alternative techniques are applied within the method of allocation to improve the condition of contiguity, by maintaining a good level of closeness between processors, to run the same task and reduce communication latency that is caused by no contiguity between the processors.

3 Basic Concepts

The proposed system is of multicomputers connected in a 2D mesh with a job queue waiting for admission to the mesh, and allowances are established as a dynamic quadratic assignment. The following definitions formally describe a system of this type.

Definition 1. An n-dimensional mesh has $k_0 \times k_1 \times ... \times k_{n-2} \times k_{n-1}$ nodes, where k_i is the number of nodes along the $i-th$ dimension and $ki \geq 2$. Each node is identified by n coordinates: $0(a), 1(a), ..., n-2(a), n-1(a)$, where

$$0 \leq i(a) < k_i for 0 \leq i < n.$$

Two nodes a and b are neighbors only if $i(a) = i(b)$ for all dimensions except for a dimension j, where $j(b) = j(a) \pm 1$. Each node in a mesh refers to a processor and two neighbors that are connected by a direct communication link.

Definition 2. A 2D mesh definition, which is referenced as $M(W, L)$ consists of $W \times L$ processors, where W is the width of the mesh and L is the height of the mesh. Each processor is denoted by a pair of coordinates (x, y), where

$$0 \leq x < W and 0 \leq y < L.$$

A processor is connected by a bidirectional communication link to each of its neighbors. For each 2D mesh $= P_{ij}$.

Definition 3. In a 2D mesh, $M(W, L)$, a sub-mesh: $S(w, l)$ is a two-dimensional mesh belonging to $M(W, L)$ with a width w and a height l, where

$$0 < w \leq w and 0 < l \leq L,$$

and $S(w, l)$ are represented by coordinates (x, y, x', y'), where (x, y) is the lower left corner of the sub- mesh, and (x', y') is the upper right corner. The node in the lower left corner is called the base node of the sub-mesh and the upper right corner is the end node. In this case $w = x' - x + 1$ and $l = y' - y + 1$. The size of $S(w, l)$ is: wxl processors.

Definition 4. In a 2D mesh $M(W, L)$, a sub-mesh available $S(w, l)$ is a sub-mesh that meets the conditions: $w \geq \beta \geq \alpha$ assuming that the allocation of

$S(\alpha, \beta)$ required where the allocation refers to select a set of processors to an arriving task.

Definition 5. Let ϑ be a set of system tasks, such that $\vartheta = J_1, J_2, \ldots, J_n$, where n is the number of tasks at time t y ϑ_k a set of sub-tasks of the task k where: $\vartheta_k = j_{k1}, j_{k2}, \ldots, j_{kf(k)}$ y $f(k)$ is the total number of sub-tasks of the task j. For each task each task j and each sub-task $f(k) \in j$ a processor $m_i \in P$ is had that should run j and sub-task $j_{kf}(k)$, consuming an uninterrupted time of $t \in \mathbb{N}$.

Definition 6. Given two matrices of size $n \times n$: a flow matrix F whose (i, j) elements represent flows between i and j tasks and an array of distances D whose (i, j) represent the distance between sites i and j. An assignment by the vector p, which is a permutation of the numbers $1, 2, ..., n$. $p(j)$ is where the task j is assigned. Thus, the quadratic assignments can be written as:

$$min_p \in \sum_{i=1}^{n} \sum_{i=1}^{n} f_{ij} dp(i)p(j).$$

Definition 7. An optimization problem, is one whose solution involves finding a set of candidate alternative solutions that best meet objectives. Formally, the problem consists of the solution space S and objective function f. Solving the optimization problem *(S, f)* it is to determine an optimal solution, namely, a feasible solution $x^* \in S$ such that $f(x^*) \leq f(x)$ for any $x \in S$. Alternative solutions can be expressed by assigning values to some finite set of variables $X = Xi : i = 1, 2, ..., n$. If U_i is denoted the domain or universe (set of possible values) of each of these n variables, the problem is to select each variable X_i domain U_i value x_i assigned that, subject to certain restrictions, optimizes an objective function f. The universe of solutions is identified with the set:

$$U = x = (x_i : i = 1, 2, \ldots, n) : x_i \in U_i.$$

The problem constraints reduce the universe of solutions to a subset of $S \subseteq U$ solutions, called feasible space.

Definition 8. Utilization. It is defined as the fraction of time in which the system was used. And it is given by:

$$U_G = W_G/(C_G * m_G),$$

where W_G is the amount of work the system, C_G is the end time of execution of all tasks in the system, m_G is the total number of processors in the system.

Definition 9. Processing Performance (throughput). The number of tasks completed per unit of time in the system, and it is given by:

$$n/C_G,$$

where n is the total number of tasks in the system.

Definition 10. Mean turnaround time. The average time it takes for all tasks upon entering the queue until their execution is ended. It is calculated as:

$$\frac{1}{n} \sum_{j=1}^{n} t_{\mathrm{t}}^{\mathrm{j}},$$

where $t_{\mathrm{t}}^{\mathrm{j}} = c^j - r^j$, c^j is the time of completion of the task and r^j is the delivery time of the task j.

Definition 11. Waiting time. It is defined as the average waiting time before starting the task execution. It is calculated as:

$$\frac{1}{n}\sum_{j=1}^{n} t_{\mathrm{w}}^{\mathrm{j}},$$

where

$$t_{\mathrm{w}}^{\mathrm{j}} = t_{\mathrm{s}}^{\mathrm{j}} - r^j,$$

where $t_{\mathrm{s}}^{\mathrm{j}}$ is the start time of execution of the task j.

Definition 12. Coefficient response (response rate). It is defined as the average of the response factors of all tasks. It is defined as:

$$\frac{1}{n}\sum_{j=1}^{n}(t_{\mathrm{w}}^{\mathrm{j}} + P^j)/P^j,$$

where P^j is the runtime and $t_{\mathrm{w}}^{\mathrm{j}}$ is the waiting time of the task j.

Definition 13. Competitive ratio. A measure of system performance defined as:

$$p = c_g/c_{LB},$$

where c_g is the time of completion and c_{LB} is the minimum time to complete tasks, calculated as: $maxw_G/m_g, t_{\mathrm{g}}^{\max}$, where $t_{\mathrm{g}}^{\max}$ the maximum runtime of the n tasks.

4 The UMDA

The EDA (Estimation of Distribution Algorithms), are evolutionary algorithms that use a collection of candidate solutions for accomplishing search paths avoiding local minimums [21,22]. These algorithms use the estimation and simulation of the joint probability distribution, as a mechanism of evolution, instead of directly manipulating the individuals that represent solutions to the problem. EDA algorithm starts randomly generating a population of individuals that represent solutions to the problem. Three types of operations are performed iteratively on the population [21,22]. The first type of operation is the generation of a subset of the best individuals in the population. Secondly, a learning process from a probability distribution model is made from selected individuals. Third, new individuals are generated by simulating model the distribution obtained. The algorithm stops when a certain number of generations are reached or when performance fails to improve significantly.

To estimate in each generation the distribution of joint probability, from selected individuals, we use the algorithm of the univariate marginal distribution (UMDA by its acronym, Univariate Marginal Distribution Algorithm). Thus, the joint probability distribution is factored as the product of independent univariate distributions [21,22], that is:

$$p_l(x) = p(x|\, D_{\text{l-1}}^{\text{Se}}) = \prod_{\text{i=1}}^{\text{n}} p_l(x_i).$$

Each univariate probability distribution is estimated from marginal frequencies:

$$p_l(x_i) = \frac{\sum_{j=1}^{n} \delta_j(X_i = x_i|\, D_{\text{l-1}}^{\text{Se}})}{N},$$

where

$$\delta(X_i = x_i|\, D_{\text{l-1}}^{\text{Se}}) = \left\{ \begin{array}{l} 0\, if\, is\, the\, i-th\ D_{\text{l-1}}^{\text{Se}},\ X_i = x_i \\ 1\, in\, other\, case \end{array} \right\}.$$

The pseudo code for UMDA proposed in [21,22] is shown in table 1.

Table 1. The pseudo code for UMDA [21,22]

$D_0 \leftarrow$ Generate M individuals (the initial population) random
Repeat for $l = 1, 2, ..$ until stop criteria
$D_{\text{l-1}}^{\text{Se}} \leftarrow$ select $N \leq M$ Individuals of D_{l-1} according to the selection method:
$p_l(x) = p(x|\, D_{\text{l-1}}^{\text{Se}}) = \prod_{\text{i=1}}^{\text{n}} p_l(x_i) \prod_{\text{i=1}}^{\text{n}} \frac{\sum_{j=1}^{n} \delta_j(X_i = x_i|\, D_{\text{l-1}}^{\text{Se}})}{N} \leftarrow$
Estimate the joint probability distribution D_l
Sample M individuals (the new population) of $P_l(x)$

In this paper, the application of UMDA evolutionary algorithm is carried out as follows:

1. A set of tasks is dynamically extracted from the queue that fit in the free submeshes, this set of tasks represents a possible assignment (individual); this process is repeated until n number of individuals (user-defined), that constitute a population.
2. For each individual in the population, the five objectives are evaluated to determine the subset of assignments (subset of the best individuals) that show results closest to maximization or minimization established for each objective function.
3. The probability distribution model learning process, is produced from selected individuals representing the best assignments to the mesh of processors.
4. A new generation of individuals occurs by simulating the distribution model obtained in the previous step.
5. An algorithm stop mechanism is activated when minimizing or maximizing of the objective functions.

During the process of evaluating each target for each of the individuals, the contrasts are shown in the results, due to the improved results from a function other objective result worsen. The following section explains through examples how the objectives are opposed.

5 Process for the Analysis of the Contrast between the Objectives of the Planning and Allocation of Tasks in a 2D mesh

To make the contrast analysis of the 5 goals, extracted from research and previously discussed in previous sections, first, the results were considered [12,13] of the trials of the UMDA that evaluates each target separately. Second, additional trials were conducted of the same algorithm to determine the contrast of each of the targets within the same group. The formal approach of the found contrasts are detailed in the following section. Upon completion of the experiments, and based on the results a scale of priorities is formed, that in which is used in this research as a determiner to find the best assignment of tasks to the processor mesh.

5.1 Opposition of the Objectives during Planning and Allocation of Tasks

In this section, the found contrasts are explained through a set of examples between the objectives that are pursued to meet the optimal utilization of the processors of the mesh. A formal approach to them is also performed.

Objectives 1 and 2, which seek to minimize the number of assignments to the mesh of processors, to minimize the time that jobs remain in the queue, is at odds with the objectives of minimizing the use of processors in the mesh and minimizing starvation of tasks. To illustrate how these four objectives are opposed, consider that at time t, the allocator 29 reports free processors (as shown in figure 1), with this information the scheduler determines that the set of the 5 tasks T_0, T_1, T_2, T_3 and T_4 are candidates to fill 21 processors in the mesh, or assign the task requiring 26 T_5 and T_4 processors requesting task 3. Assign the set of 5 tasks releases the same number of positions in the queue for the entry of new tasks, thus reducing the number of accesses to the queue to find more tasks, this allows a greater number of users and tasks to be served and the waiting time of tasks in the head of the queue is decreased; but in opposition to this, an external fragmentation of 8 processors is generated and starvation in a cycle of tasks increases, upon not being served a task that requires a large number of processors.

Now, if the T_4 and T_5 tasks assigned do not produce starvation nor external fragmentation, a smaller number of tasks can be accepted in the queue, so the number of assignments to the screen increases and therefore so do the time tasks must wait to enter the mesh of processors.

Objective 3, which seeks to maximize the use of the processors in the mesh, contrasts with objective 5, which maximizes the adjacency of the occupied processors in the 2D mesh. To illustrate the contrast between these objectives, consider the example above. By searching for the lowest communication cost, all of the 5 selected tasks: T_0, T_1, T_2, T_3 and T_4 are assigned in contiguous processors, as follows: T_0 task is assigned to the sub-mesh $< 4, 0 >< 5, 2 >$ regardless of the

processor to $< 4, 2 >$, the T_1 task is assigned to the sub-mesh $< 2, 0 >< 3, 1 >$, the task T_2 is assigned to the sub-mesh $< 0, 5 >< 2, 6 >$ regardless of the processor to $< 2, 5 >$, the task T_3 is assigned in submesh $< 0.2 >< 1, 3 >$, and T_4 task is assigned to the sub-mesh $< 6, 3 >< 7, 4 >$ regardless of the processor to $< 7, 4 >$. If the proposed method detects the increase of starvation in the system, the T_5 task will be assigned to mesh with the task T_1 or to task $T3$ that is selected to occupy all of the processors, after a search in the queue and to avoid external fragmentation; in this way the 29 free processors in the mesh will remain busy. If targets 3 and 5 are opposed one can deduce that to assign the task T_1 or T_3 and T_5, the use of processors in the mesh is maximized, but the adjacency between processors is minimized, and in contrast, if the set of 5 tasks is assigned, the adjacency between processors is maximized, but an 8 processor external fragmentation occurs.

Objective 1 minimizes allocations to the mesh of processors, runs counter to *objective 5*, which maximizes the adjacency between processors. The contrast between these two objectives appears when you intend to assign a large number of tasks in the mesh of processors, and processors to which tasks are assigned are not close enough together or contiguous, to avoid producing very high communication costs. If we consider assigning the set of 5 tasks, T_0, T_1, T_2, T_3 and T_4, the number of assignments made to the mesh is minimized, but if the positions of the free processors in the mesh are adjacent, occur tasks will be assigned to the the mesh in a very disjointed way, causing the adjacency of processors to be minimal and communication costs between tasks to be very high.

Objective 2 which seeks to minimize the waiting time of tasks in the queue, runs counter to the objective of maximizing the adjacency between processors. Using the same example in the previous section and considering that the *objective 2*, sets to minimize the waiting time of tasks in the queue, upon assigning the largest number of tasks in the mesh the wait time fora set of tasks is minimized. In the allocation that occurs at time *t*, fewer jobs wait in a queue. For such cases if two objects are affected simultaneously by a third objective, and we manage to improve them, the algorithm will decide, based on this option when planning and allocating, considering the generated cost involved in external fragmentation and increased task starvation.

Objective 4, which aims to minimize the starvation of tasks, runs counter to the objective 5, which maximizes the adjacency between processors. The decision to allocate a greater number of tasks to minimize waiting times and maximize the use of processors, seems like a viable option, but if we consider a third goal in conflict that tries to reduce task starvation in the system, we find then that we are in favor of two objectives (2 and 3), and sacrifice also 2 (objectives 4 and 5).

Based on the explained examples, we have found that maintaining a strict control of tasks that fit in the mesh, to meet the proposed objectives, produces exhausting searches in the queue, and calculations to locate tasks in the best position in the mesh [12]. Rather than seeking the best positions of tasks in the mesh, you should perform an analysis of the objectives, which seek to optimize,

because when trying to locate submeshes of sizes that the tasks required for the sole purpose of being contiguous, without considering a evaluation of other objectives, can lead to poor results in response times and system performance.

Based on the above explanations, in this paper a scale of priorities for the objectives is made, to be considered during the planning and allocation of tasks in multicomputers systems. It is noted that this scale of priorities, is considered in the algorithm proposed in this paper, and with this the results explained in the section of results were obtained.

6 Scale of Priorities of the Objectives

This paper presents a stratification of the proposed objectives, based on the results obtained with the algorithm UMDA and observations made in the above experimentation have been performed in order to determine the best allocations, should similar or identical l values be found when compared to those of objective's rankings.

Stratification proposal is as follows:

1. Objective 5, which sets to maximize adjacency between processors, is considered the major goal in the allocation for five situations that arise during task assignments to the mesh of processors, in experiments: the first factor to consider is the communication time tasks consume during execution, because the non-adjacent processors generate very high communication costs and even more when to perform tasks that require large quantities of processors within the mesh. Although the search for free submeshes is a tedious process and consumes time from the processor, it is a task that should be extensive at any time.
2. Objective 3, maximize the use of processors to reduce external fragmentation, is considered secondly because of its importance in the allocation of the processors in the mesh. It is importance, is that it serves as a support for Objective 1, the experiments conducted allow us to observe that in order to maximize the use of processors, allocations should be made in the greater number of processors that are adjacent within the mesh of processors. To meet this objective, the algorithm that makes finding free submeshes must be big enough.
3. Objective 4, minimize task starvation, it is located on the third level of importance, because being able to meet the two prior objectives this allows a safe handling of tasks, that require large numbers of processors in the mesh, thus avoiding task starvation. If the free search algorithm submeshes provides sets of adjacent free processors, it is possible to avoid task starvation upon placing them within the mesh.
4. Objective 1, minimize the number of assignments to the mesh of processors, i.e. minimize the number of planifications, that the algorithm must perform with the tasks that remain in the mesh, it is considered as the fourth objective of importance to the evaluation and is considered to include Objective 2,

because achieving being able to minimize the number of assignments, to the mesh of processors reduces the time that tasks waiting in the queue

5. Objective 2 is considered a level 5 of importance, which sets to decrease the waiting time of tasks in the queue.

7 Results

The experiments were performed with different workloads in the Liebres Inteligentes [14] system and with different sizes in the queue. Size loads of 256, 512, 1024 and 2048 tasks are considered in the system. The lengths of the queue are carried out in 10, 20, 30 and 50 tasks with their respective subtasks. The number of subtasks for each task is 255 at most, considering that the size of the mesh of processors is $< 16 \times 16 >$. In figure 2, the results obtained for each objective function are shown. For reasons of space, only the loads on the system and the values obtained for the objective function are shown, up to 800 tasks.

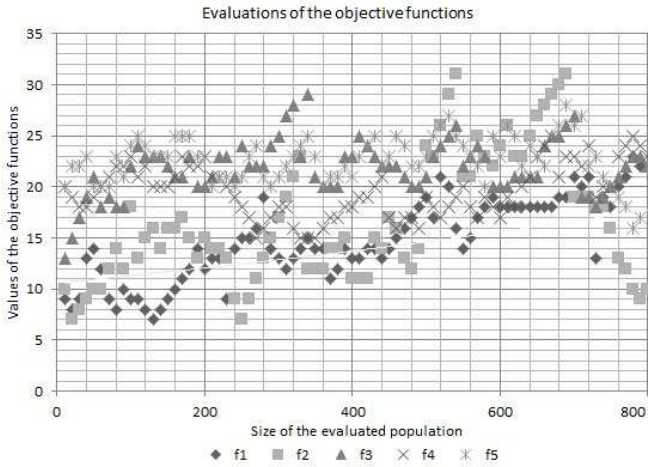

Fig. 2. Experiment 1, the chart shows the values obtained for the objective functions

In the X coordinate, different system loads are shown and in the Y coordinate obtained values of the objective functions are given. The performance of the functions, shown at the bottom of the graph, identify the symbol used for each function, the order of the functions is ascending, not by degree of importance.

In the following paragraphs the results for each objective, and the impact they have over the functions of planning and allocation of tasks are explained.

The values for function 5, maximize adjacency between processors that seek to assign tasks with the highest degree of contiguity, show that with values of less than 100 task workloads, no acceptable values are had, but as the number of task execution increases, values significantly improve. The function is observed in this part, the number of subtasks processed exceeds the average, ie exceeds 128 subtasks task.

Function 4, minimizing starvation task has a high tendency, because the tasks that are to be processed contain a large number of subtasks, causing fewer tasks with subtasks to be addressed quick through the system. As large tasks are evicted, starvation tends to stabilize at acceptable levels causing a greater fluency in job processing.

The 3rd function, maximizes the use of processors, considered one of the most important functions for the system, it has a tendency to group values, with the results of the function 5, but as the number of tasks to be processed increases, there is a dispersion in a middle point whose tendencies show that upon obtaining higher adjacency the use of processors decreases, especially when the system starts to process jobs with a large number of subtasks.

Function 2, decreases the time that a task expected to be attended in line, shows a very clear trend, when tasks are processed with fewer resource requirements, waiting times are very short, due to the planning carried out by the algorithm . Otherwise, when the tasks are processed containing a large number of requirements, waiting times are higher, which undoubtedly, upon increasing the number of resources, this trend is easily improved. This is shown in the graph when loads between 200 and 400 jobs are processed; the values of the functions are fired very easily.

Finally, function 1, which seeks to minimize the number of assignments that the algorithm performs, shows very poor booting trends, but as the implementation progresses, their values are significantly improved. It´s tendency with the values of the function 2, makes it a dependent function that takes a curve to the values acquired in function 2. As function 2 has better values that represent a decrease of time a task has to wait to be served in the queue, the number of assignments that the allocator algorithm performs is substantially reduced. Function 2 in conjunction with function 1, has a high degree of importance on the results that the system puts out, so it is important to consider them as priorities.

Figure 3 shows another exemplifying embodiment of the system. With a greater number of tasks in the system, value trends are similar to the previous graph. With this example, it is intended that the functions obtain values that correspond to a greater load in the system. The analysis that is done in this experiment, allows us to observe that the functionality of the planning algorithm is feasible when used with heavy system loads.

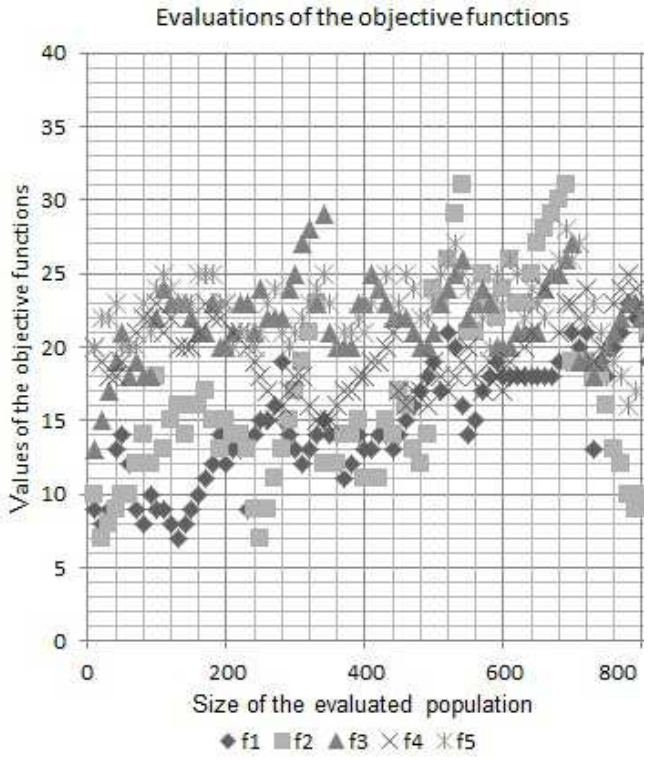

Fig. 3. Experiment 2, the chart shows the values obtained for the objective functions

8 Conclusions

Multicomputers systems are a viable option for parallel processing, because of their growth in terms of computing power and distributed storage. The inherent problems associated with their architecture are the planning and allocation of tasks. For allocating tasks to the mesh of processors many techniques based on different strategies have been proposed through: geometric figures that move across the screen to locate the free submeshes, adjustments to free submesh application sizes and techniques based on random assignments. Most of these techniques make use of planning policies, based on the first to arrive is the first to be served (FIFO First Input, First Output), that is, pre-planning is not used in the queue, furthermore it only seeks to solve one problem: the one that deals with adjacent or contiguous allocation to be able to reduce message passing between tasks and subtasks.

The method presented in this document deals with the problem of planning

and allocation of tasks on a multicomputer system as a multiobjective problem, conducting an analysis of how each goal impacts system performance, by means of using an evolutionary algorithm. The objective of this analysis is to show the values that functions present when goals are opposed, both in the planning and allocation of the mesh processors.

With the obtained results using the Liebres Inteligentes multicomputer system it is possible to deduce that in order to evaluate an allocation technique of processors in a mesh, it is necessary that this technique evaluates at least 5 different objectives, because in this way, you can determine that not only one problem will be solved, but a set of values that balance a solution will be had. An approach that seeks to solve only one objective, is not feasible, for example, one that seeks to solve task adjacency and allows system response time is not considered in the formulation of the solution.

Finally, it is very important to mention the work that is to be performed by the free submeshes search algorithm, within the mesh of processors, because it is what supports, monitors and fulfills the most important objectives within the proposed stratification in this paper research.

References

[1] Grama A., Gupta A., Karypis G., Kumar V.: Introduction to Parallel Computing. Second Edition. Addison Wesley (2003)

[2] Bani S., Ababneh I., Ould M.: A Performance Comparison of the Non-Contiguous Allocation Strategies in 2D Mesh Connected Multicomputers. International Conference On Communication, Computer And Power (ICCCP'09) MUSCAT (2009)

[3] Ababneh I., Bani-Mohammad S.: A new window-based job scheduling scheme for 2D mesh Multicomputers. Simulation Modeling Practice and Theory 19, pp. 482–493 (2011)

[4] Adiga N.R., Almasi G., Almasi G.S., Aridor Y., et al.: An Overview of the BlueGene/L Supercomputer. Team IBM and Lawrence Livermore National Laboratory.

[5] Bokhari. S. H.: Communication Overhead on the Intel PARAGON, IBM SP2 & MEIKO CS-2. [Online] http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19960004071.pdf

[6] Ahmad S. E.: Processor Allocation with Reduced Internal and External Fragmentation in 2D Mesh-based Multicomputers. Journal of Applied Science 11(6), pp. 943–952 (2011)

[7] Das D., Pradhan D. K.: Job Scheduling in Mesh Multicomputers. IEEE Transactions On Parallel And Distributed Systems, 9(1) (1998)

[8] Foster I.: Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering. Addison Wesley (1995)

[9] Heiss H. U.: Dynamic Partitioning of Large Multicomputer Systems. Proc. Int. Conf. on Massively Parallel Computing Systems (IEEE MPCS94), Ischia (1994)

[10] Deb K.: Multi-Objective Optimization Using Evolutionary Algorithms. Wiley (2001)

[11] Velarde A., Ponce de Leon E., Díaz E., Padilla A.: Planning and Allocation of processors in 2D meshes. Doctoral Consortium. Mexican Internacional Conference on Artificial Intelligence MICAI 2010 Pachuca Hidalgo, México (2010)

[12] Velarde A., Ponce de Leon E., Díaz E., Padilla A.: Dynamic quadratic Assignment to Model Task Assignment Problem to Processors in 2D Mesh. Advances in Soft Computing Algorithms. Research in Computing Science, Vol. 54, pp. 199–218 (2011)

[13] Velarde A., Ponce de Leon E., Diaz E.: Planning and Allocation Tasks in a Multicomputer System as a Multi-objective Problem. Advances in Intelligent Systems and Computing 227. EVOLVE 2013, International Conference, Leiden, The Netherlands, Springer (2013)

[14] Velarde A.: Liebres InTELigentes: Sistema de Multicomputadoras para la enseñanza de la programación de los sistemas de cómputo de alto rendimiento en Instituciones de Educación Superior. Articulo aceptado en: Congreso Internacional de Investigacion en Ciencias y Sustentabilidad de AcademiaJournals, Tuxpan, Ver., Mexico (2015)

[15] Lo V., Windisch K., Liu W., Nitzberg B.: Non-contiguous processor allocation algorithms for mesh-connected multicomputers. IEEE Transactions on Parallel and Distributed Systems, vol. 8, no. 7, pp. 712–726 (1997)

[16] Chang C.Y., Mohapatra P.: Performance improvement of allocation schemes for mesh-connected computers, Journal of Parallel and Distributed Computing, vol. 52, no. 1, pp. 40–68 (1998)

[17] Bani A. S.: Submesh Allocation in 2D Mesh multicomputers: Partitioning at the Longest Dimension of Request. The International Arab Journal of Information Technology, Vol. 10, No.3, pp. 245–252 (2013)

[18] Procsimity V4.3 User's Manual. University Oregon (1997)

[19] Zolfaghari R.: Efficient Algorithm for Processor Allocation in Mesh Multicomputers Network with Limitations and Assumptions. IJCEM International Journal Of Computational Engineering & Management, Vol. 16, no. 4 (2013)

[20] Suzaki K., Tanuma H., Hirano S., Ichisugi Y., Connelly C., Tsukamoto M.: Multi-tasking Method on Parallel Computers which Combines a Contiguous and Non-contiguous Processor Partitioning Algorithm. Proceedings of the 3rd InternationalWorkshop on Applied Parallel Computing, Industrial Computation and Optimization, Lecture Notes in Computer Science, Springer, London, pp. 641–650 (1996)

[21] Larrañaga P., Lozano J. A., Mühlenbein H.: Algoritmos de estimación de distribuciones en problemas de optimización combinatoria. Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial (2003)

[22] Lozano J. A., Larrañaga P.: Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation. Kluwer Academic.