Empirical Study of Tabu Search, Simulated Annealing and Multi-Start in Fieldbus Scheduling

Lucy María Franco Vargas
Universidade Federal de Santa Catarina
Caixa Postal 476 – 88.040-900 –
Florianópolis – SC – Brasil
lucymfv@gmail.com

Romulo Silva de Oliveira
Universidade Federal de Santa Catarina
Caixa Postal 476 – 88.040-900 –
Florianópolis – SC – Brasil
romulo@das.ufsc.br

Abstract

The Foundation Fieldbus specification is a technology that allows to construct control strategies in terms of blocks diagrams, distributed among the field devices on the network. First, the control engineer defines the control strategy to be used and then it is done the allocation (in the different field devices on the network) of the function blocks used in the defined strategy. After these allocation it is necessary to do the scheduling of the function blocks to guarantee a correct order of execution and communication. This paper aims at evaluating the application of three metaheuristic methods for the resolution of the allocation and scheduling of function blocks problem in fieldbuses that follow the Foundation Fieldbus specification. The used methods are the Multi-Start, the Simulated Annealing and the Tabu Search.

1. Introduction

The traditional communication architecture for control systems is point to point. However, a traditional centralized point to point control system is no longer suitable to meet new requirements such as modularity, decentralization of control, integrated diagnostics, quick and easy maintenance, and low cost [1]. Network systems with common bus architecture, called Networked Control Systems (NCS), provide several advantages such as reduction of wiring, distributed processing, ease of system diagnosis and maintenance, and increased system agility [2].

Foundation Fieldbus (FF) is an all-digital, serial, two-way communication system that allows the communication between control and supervision equipments and field devices [3]. The FF specification is not only a communication protocol but also a programming language for building control strategies [4]. The FF user layer is based on blocks distributed among the devices on the network. The blocks are representations of different types of application functions. One type of those blocks is the function block, which is used to build the control strategy that will allow to control the industrial plant.

After the control strategy being defined in terms of function blocks, it is necessary to do the allocation of each block to the several devices on the network. The allocation refers to the decision of which block executes on which field device. Although the manual allocation is feasible in small systems, it becomes unfeasible in larger plants, with dozens or even hundreds of sensors and actuators.

After the allocation of the function blocks on the respective devices it is necessary to do the scheduling of these blocks and the messages sent between function blocks. The scheduling refers to the decision of "which functional block executes when on the processor that it was allocated" and also "which message is transmitted when on the communication bus".

Problems like these are difficult to solve with exact methods. Although a method that solves them exists, its computational cost will be very high, what will turn it unfeasible. This type of problem is classified as NP-Hard or NP-Complete by the computational complexity theory [6]. For the resolution of this kind of problem one can use methods known as metaheuristics. In [8], a metaheuristic is defined as a higher level heuristic procedure designed to guide other methods or processes towards achieving reasonable solutions for difficult combinatorial mathematical optimization problems. These methods are particularly concerned with not being trapped in a local optima (for problems that have many local optima) and/or to reduce the space of search in a reasonable way. Metaheuristics provide a general framework to create new hybrid algorithms combining different concepts derived from the artificial intelligence, the biological evolution and statistical mechanisms. Some of these metaheuristic methods are: Multi-Start, Simulated Annealing and Tabu Search.

In this paper it is made an empirical study of the Tabu Search, Simulated Annealing and Multi-Start metaheuristics when applied to the function blocks and periodic messages scheduling of the FF network.

The relative merit of the metaheuristics largely depends on the considered sets of problem instances. This paper uses in the experiments a highly representative set of the instances of problems that appear in the context of the Foundation Fieldbus.

In section 2 it is defined the FF system, whose utilization is assumed for the communication network. Section 3 presents the description of the problem that is studied in this paper. In the next section it is presented the three metaheuristics that will be used in this paper. In section 6 it is made the definition of the neighborhood and the solution quality. Finally, a set of applications shows the process of allocation and scheduling of function blocks and periodic messages and the obtained results after the application of the proposed metaheuristics. Section 7 contains the conclusions.

2. The Foundation Fieldbus (FF) Standard

FF defines a serial communication protocol. There are two subsystems in FF, the H1 that interconnects field devices and the HSE (High Speed Ethernet) that provides integration of high speed controllers, H1 subsystems, data servers and workstations [3].

The control of the communication and the access to the fieldbus is responsability of a device called Link Active Scheduler (LAS). The LAS manages the permission given to the devices that want to initiate the transmission of data onto the bus, so that only one device accesses the bus at each instant.

The communication on the bus can happen in two ways. The scheduled communication is used to transfer data that have some kind of timing constraints, such as those used in control strategies. The unscheduled communication is used to transfer data with no timing constraints. This paper only considers the scheduled communication.

The Fieldbus Foundation has defined a standard User Application Layer, based on blocks and device descriptions. The types of blocks used in User Applications are: resource block, transducer block and function block. Devices are configured using resource blocks and transducer blocks. The control strategy is built using function blocks.

When the system is configured and the function blocks are linked, a schedule is created for the LAS. Each device maintains its part of the schedule known as Function Block Scheduling, that indicates when the function blocks in the device will be executed. The scheduled execution time for each function block is represented as an offset from the beginning of the macrocycle start time. The macrocycle is the cyclic period of the control strategies.

3. Allocation and Scheduling Problem Description

A control strategy, defined in terms of block diagrams, will be executed in a distributed system composed by **m** field devices connected through a communications network in a bus format (Foundation Fieldbus network). The field devices and the bus will be considered as processors.

The function blocks (FB) are executed in the field device processors and the bus processor transmits the periodic messages sent between blocks in different devices. The execution of the function blocks and the transmission of periodic messages, will be called FB tasks and communication tasks, respectively.

A control strategy has **n** control loops wich are executed on **m** field devices.

The control loop **L[i]**, i = 1, ..., n, has $\mathbf{nb_i}$ FB tasks and $\mathbf{np_i}$ communications tasks.

A task $\mathbf{T_j[i]}$, j = 1, .., (nbi + npi), (FB or communication) of the control loop L[i] can be defined as: $T_j[i] = (C_j[i], P_j[i], D_j[i])$, where $C_j[i]$ is the computation or transmission time required for the execution of the task $T_j[i]$, the period $P_j[i]$ is the fixed time in which the task $T_j[i]$ should be executed repeatedly and the deadline $D_j[i]$ is the maximum time that the task $T_j[i]$ has to be executed. These times, $C_j[i]$, $P_j[i]$ e $D_j[i]$ are known; the $C_j[i]$ of the messages that are sent between function blocks in the same device is considered equal to 0. The deadline $\mathbf{D_j[i]}$ is considered equal to the period $P_j[i]$ of the task.

Some of the tasks $T_j[i]$ of the control loop L[i] have precedence relations. This relations will be represented by the symbol $\rightarrow$. A precedence relation is transitive, that means that if $T_j[i] \rightarrow T_k[i]$ and $T_k[i] \rightarrow T_h[i]$, then $T_j[i] \rightarrow T_h[i]$.

Each control loop L[i] has a period **P[i]** and it is the same for all the tasks (FB or communication) in that control loop; likewise, L[i] has a deadline **D[i]** , which is assumed to be equal to its period **P[i]**. Besides, the minimum release time $\mathbf{J_j^-[i]}$ for each task $T_j[i]$ of the control loop L[i] is known.

Each task $T_j[i]$ in the control loop L[i] will be scheduled $\mathbf{ns_j[i]}$ times in the macrocycle:

$$ns_j[i] = macrocycle / P[i], \quad (1)$$

where P[i] is the period of the task $T_j[i]$.

The activation $\mathbf{T_j[i][f]}$ is defined as the activation f (scheduling) of the task $T_j[i]$ in the control loop L[i] in the macrocycle, and it is described as:

$$T_j[i][f] = (A_j[i][f], C_j[i], D_j[i][f]). \quad (2)$$

$\mathbf{A_j[i][f]}$ is defined as the arrival of the activation f of the task $T_j[i]$ of the control loop L[i]:

$$A_j[i][f] = (f - 1) * P[i], f = 1, .., ns_j[i]. \quad (3)$$

The minimum release time $\mathbf{J_j^-[i]}$ for each task $T_j[i]$ of the control loop L[i] of the activation f of the task $T_j[i]$ of the control loop L[i] is:

$$J_j^-[i][f] = \begin{cases} \max\{(J_k^-[i][f] + C_k[i])\}, \\ \quad \text{if } \exists T_k[i][f] / T_k[i][f] \rightarrow T_j[i][f], k \neq j \\ \quad \text{and } T_k[i] \in L[i] \\ (f-1) * P[i], \text{ otherwise} \end{cases} \quad (4)$$

$\mathbf{D_j[i][f]}$ is the deadline of the task $T_j[i]$ in its activation f in the macrocycle:

$$D_j[i][f] = f * P[i], \qquad f = 1, .., ns_j[i]. \quad (5)$$

When there is only one control loop per network, the macrocycle will be equal to the period of the loop. If there are more than one control loop in the network and these loops have different periods, the macrocycle will be equal to the least common multiple (LCM) of the periods of all control loops.

An important aspect to be considered before doing the function blocks and periodic messages scheduling is the allocation of these FB on the devices that form the distributed system on which the control strategy will be executed. This allocation is static and doesn't change during the execution of the control strategy. The execution time of the control strategy will depend on the allocation of the FB of that control strategy because different allocations of the blocks of that strategy can result in different execution times, some higher than others.

A scheduling solution in this context consists of the definition of the allocation and also on the construction of a time grid informing which task (FB or communication) executes at each instant. The time grid will have the size of the macrocycle.

As an example consider the cascade control loop of figure 1, with five FB (AI1, AI2, PID1, PID2 and AO) to be executed on three field devices and six periodic messages (Msg1, Msg2, Msg3, Msg4, Msg5 e Msg6) that are sent between function blocks.

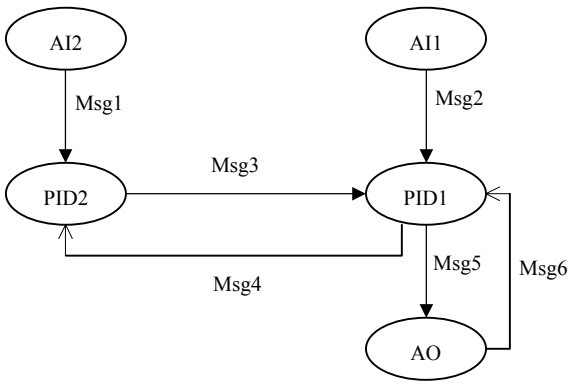

Figure 1. Cascade control loop.

Before the scheduling of the FB and the messages, it is made the allocation of these blocks on the three devices; that allocation will determine which messages will be transmitted on the bus and which will have its transmission time equal to 0. A possible allocation could be the one shown in figure 2. With this allocation the messages Msg2, Msg3 and Msg4 will be transmitted on the bus and the other three messages won't be considered for the scheduling because they will have a transmission time equal to 0 (messages between blocks on the same device).

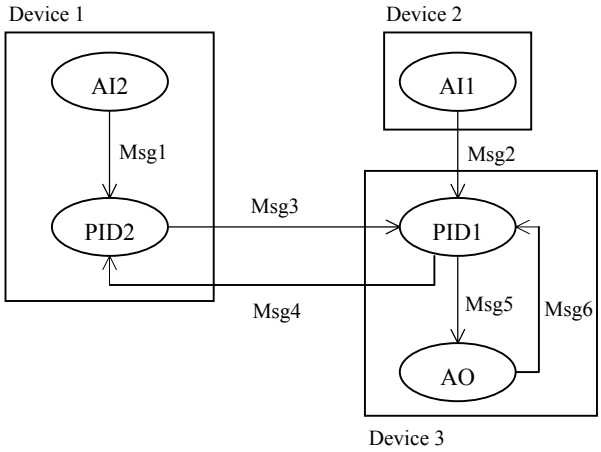

Figure 2. Function block allocation.

Table 1 shows the time parameters of the function blocks and the messages of the control loop considered in this example. Because there is just one control loop, the macrocycle will be equal to the period of the loop, 300ms.

Table 1. Time parameters of the BF and messages

BF and messages	$C_j[i]$ (ms)	$\mathbf{J_j^-[i]}$ (ms)	$\mathbf{D_j[i]}$ (ms)	P[i] (ms)
AI2	30	0	300	300
PID2	65	30	300	300
Msg3	20	95	300	300
AI1	35	0	300	300
Msg2	20	35	300	300
PID1	65	115	300	300
Msg4	20	180	300	300
AO	30	180	300	300
Msg1	0	30	300	300
Msg5	0	180	300	300
Msg6	0	210	300	300

Considering the precedence relations and the time parameters of table 1, a possible FB and messages scheduling it is shown in figure 3. As it can be seen in figure 3, the maximum time of execution of the control strategy as a whole is 210ms, but with a different allocation of the FB this value would change.

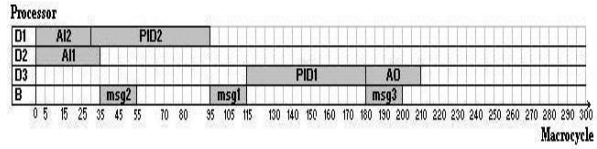

Figure 3. FB and messages scheduling.

4. Compared Metaheuristics

In this section it will be described the three metaheuristics that were considered in this study.

4.1. Multi-Start

The Multi-Start method have two phases, the first one in which the solution is generated and the second one in which the solution is typically (but not necessarily) improved. Each global iteration produces a solution (usually a local optimum) and the best overall will be the algorithm's output. The pseudo-code of the Multi-Start procedure is presented below. [7].

```
Initialise i=1
While (Stopping condition is not satisfied)
{
   Step 1. (Generation)
      Construct solution x_i.
   Step 2. (Search)
      Apply a search method to improve x_i.
      Let x_i' be the solution obtained.
   If (x_i' improves the best) then
      Update the best.
   i=i+1
}
```

4.2. Tabu Search

Tabu Search (TS) is a metaheuristic that guides a local search heuristic to explore the solution space so it avoides to be trapped in a local optimum. It is based on general principles of the artificial intelligence (AI). It takes from the AI the concept of memory and implements it through simple structures with the objective of driving the search considering its history.

The method begins with a complete, feasible solution and, just like local improvement, it continues developing additional complete solutions from a sequence of neighborhoods. However, to escape from a local optimum, moves to neighbors with inferior solutions are permitted [8].

Solutions visited recently are labeled as tabu and maintained in a list denominated Tabu List to prevent that certain solutions happen in a τ number of iterations, called size (or length) of the list. This size is a key and controllable parameter of the metaheuristic. After this number of iterations it is considered that the search is in a different area and the old solutions can be released from the tabu state (the solutions are removed from the Tabu List and therefore eliminated its tabu state). However, the tabu state of a solution can be canceled (before finishing the τ number of iterations) by the use of the denominated aspiration criterion. This criterion can be defined as those conditions that, if they are satisfied, they would allow to reach a solution although it has tabu state [8]. Tabu Search's algorithm is presented below.

```
Begin
  Generate an initial solution s
  TabuList ← 0
  While (Stopping condition is not satisfied) do
     s ← ExtractBestElement(Neighborhood(s)\
                                    TabuList)
     Update(TabuList)
  End While
End
```

The TS algorithm is based on the interaction between the short term memory and the long term memory [8]. The short term memory gets used to store attributes of solutions recently visited and its objective is to thoroughly explore a given area of the solution space. The second type of memory, long term, stores the frequencies or occurrences of attributes in the visited solutions trying to identify or to differentiate areas.

4.3. Simulated Annealing

This method simulates the annealing process used in metallurgy, where the metal is cooled in appropiate conditions and a simple crystal can be obtained. In the annealing the metal is heated to high temperatures, causing a violent shock in the atoms. If the metal is cooled in an abrupt way, the microstructure tends to a randomly unstable state and if it is cooled in a sufficiently slow way, the system will find an equilibrium point characterized by an orderly and stable microstructure. The project variables are randomly perturbed and the best value of the objective function is stored at each perturbation. The temperature is then reduced and new attempts executed. This procedure continues until we escape from a local optimum. At the end of the process it is possible to be obtained a global optimum. The distinctive characteristic of the algorithm is that it incorporates random jumps for potential new solutions. This ability is controlled and reduced as the algorithm progresses. More information can be found in [5].

A random starting point is chosen, and the energy, E_s, evaluated. A random point in the neighborhood space is then chosen, and the energy, E_n, evaluated.

This point becomes the new starting point if either $E_n \leq E_s$, or if: $e^x \geq random(0,1)$. Where $x=(E_s - E_n)/C$; C is the control variable.

The control variable C is analogous to the temperature factor in a thermodynamic system. During the annealing process C is slowly reduced, making higher energy jumps less likely. Eventually, the system "freezes" into a low energy state. The structure of the algorithm is shown below [10].

Chose a random starting point P_0
Chose a starting temperature C_0
Repeat
  Repeat
    E_P := Energy at point P_n
    Chose T, a neighbor of P_n
    E_T := Energy at point T
    If $E_T < E_P$ **then**
      $P_{n+1} = T$
    Else
      $x = (E_P - E_T) / C_n$
      If $e^x \geq random(0,1)$ **then**
        $P_{n+1} = T$

      Else
        $P_{n+1} = P_n$
      End
    End
  Until thermal equilibrium
    $C_{n+1} = f(C_n)$
Until some stopping criterion

5. Definition of Neighborhood and Quality of Solution

An important subject is the definition of the neighborhood of a point (or solution). The choice of the neighborhood can vastly affect the performance of the algorithm. While choosing a neighborhood containing a vast number of candidates solutions will increase the probability of finding good solutions, the computation time required to do the selection of the neighbors will also increase.

In this work, a solution is formed by function blocks and messages with its respective start times of execution and transmission and the allocation of the blocks in the different processors. Formally, be X the set of solutions of a problem. Each solution $x \in X$ has a set of solutions associated to it $N(x) \subseteq X$, that will be denominated neighborhood of x. And, given a solution x, each solution from its neighborhood, $x' \in N(x)$, can be obtained directly from x through an operation called movement.

For this work we defined three types of neighborhoods. In the first one, two function blocks are randomly selected and it is made a change of processors between them. Besides, to each activation of those blocks is attributed a start time of execution that is randomly chosen between the beginning of the period (arrival) and the deadline of that activation.

After these alterations, are established the start times and the time of transmission of the messages sent between the function blocks that form the control strategy, be these external messages (between blocks in different processors) or internal (between blocks in the same processor). For this, firstly it is verified if the predecessors and successors blocks of each one of the messages are in the same processor or not. If they are not, and the end of execution of the predecessor block is posterior to the start time of execution of the successor block, the start time of the message will be the same to the end of execution of the predecessor block. Otherwise, if the end of execution of the predecessor is previous to the successor's start time, a random value is selected between these two values to be the start time of execution of the message (external). If they are in the same processor the start time of execution of the message (internal) will be the the end of execution of the predecessor block. The Neighborhood 2 corresponds to a variation of the Neighborhood 1, whose results are similar.

In the Neighborhood 3 a message is randomly chosen and a processor that is attributed to the successor block of this message. After that, it is established the start times and times of transmission of the messages (as they were made in the Neighborhood 2). Then, for each activation of the successor block of the chosen message, it is attributed a start time of execution randomly selected between the end of execution of the message and the deadline of the activation of this block.

To evaluate the quality of a solution, it is done a calculation based on the amount of conflicts between function blocks, conflicts on the bus, the ignored precedences and the deadlines that are not respected. Conflicts between blocks happen because a function block can not begin its execution if another block is executing on the same processor. Conflicts on the bus happen because a message can't be transmitted on the bus if another message is already being transmitted. It can be said that a precedence is ignored when a predecessor block finishes its execution after the beginning of the message or the successor block begins its execution before the end of the message transmission. It is considered that the deadline of a function block is not respected, if the block finishes its execution after this deadline. The same verification is made for the deadlines of the messages.

It is considered as the best solution the one that presents the smallest value for the result of the formula below. The constants in the formula were used so as the metaheuristic gives priority to those conflicts with the largest value of the constant and therefore solves them first than the other ones. These values for the

constants were defined after a set of preparatory experiments.

*Evaluation = (100 * NumberIgnoredPrecedences) + (1*NotRespectedDeadlines) + (10* BusConflicts) + (1*BlocksConflicts).* (5)

It will be considered as a satisfactory solution the one that gives as a result the value 0 in the same evaluation formula.

6. Experiments

Several experiments were accomplished with the objective of evaluating the behavior of three metaheuristics in the allocation and scheduling of control strategies (function blocks and periodic messages) as described in previous sections.

Function blocks that only can execute in specific processors were considered (for example, the analogic input AI and the analogic output AO blocks) and blocks that doesn't have a pre-defined processor (for example, the controller block PID). The transmission time of a message through the bus was defined as being 30ms. The computation time of each function block depends on the function that it implements. For the experiments, to each block it is attributed a random computation time that varies between the minimum value of 10ms and the maximum value of 40ms for blocks that have to be executed in specific processors (AI, AO, etc.) and between the values 40ms and 100ms for those blocks that can execute in any processor (PID, etc.).

The used control loops vary from the simplest to the more complexes to generate the different computational loads. The definition of the computational load tried to use realistic configurations and values, obtained from the literature. The characteristics of these loops are described in table 2. Figures 4 and 5 show one of the applications of table 2 and its control loop using a typical notation of FF.

In the accomplished experiments it was considered that the deadline of the control loops is equal to its period. Besides, the first three loops of table 2 are called as small loops and the others as big loops.

Table 2. Control loops characteristics [9][11]

Application	Number of blocks	Number of processors	Num. of messages
Cascade control	5	3	6
Ratio control	5	3	5
Rate control loop with lead-lag	5	3	5
Flow compensation configuration, with totalization	6	4	5
Hydrostatic tank gauging	7	3	8
Combustion control with double cross limits	9	5	15
3 Element boiler level / Feed water control	8	4	9
Temperature cascade control	7	4	9
Control example	6	4	6

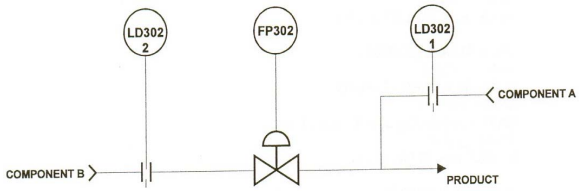

Figure 4. Application: Ratio Control.

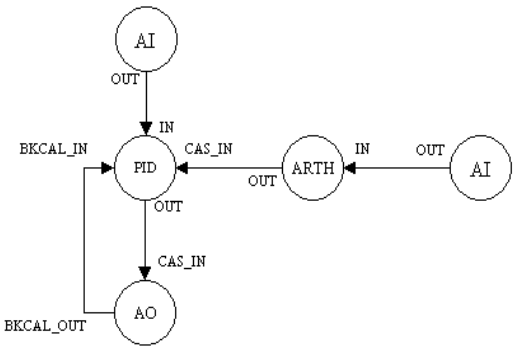

Figure 5. Control Loop: Ratio Control.

Three types of computational loads were considered, called simple, medium and complex. They are randomly generated by selecting the loops (of table 2) that will form each one of these types of load.

The simple load is formed by 2 control loops, being 2 small or 1 small and 1 big. This load can have a maximum of 14 function blocks, 8 processors and 21 messages. The medium load is composed of 4 control loops (2 small and 2 big) and can have up to 28 function blocks, 16 processors and 42 messages. The complex loads are formed by 5 control loops (2 small and 3 big). This type of load can have up to 37 blocks, 21 processors and 57 messages.

The quality of the final solution given by each one of the metaheuristics used in this work and also the computational cost necessary to reach this solution depends on the good choice of certain parameters.

The Multi-Start algorithm always starts from a solution randomly generated and it makes a greedy non-exhaustive search in the neighborhood. After a certain number of attempts without improvement of the solution it interrupts that search and begins again, starting from other solution randomly generated. The

number of attempts without success before abandoning a "start" was empirically defined through an initial period of refined adjustment, during which statistics were not collected. The selected value for this parameter allow that the metaheuristic evaluates a great quantity of neighbors (if not all) of the neighborhood before trying in a new starting point.

As it was seen, the Tabu Search uses a list where it maintains the solutions (or attributes of the solutions) that were recently visited to prevent that certain solutions are repeated in a number x of iterations (size of the list). The tabu state is attributed to the solutions in the list, which can only be cancelled for a certain solution if it is better than the current solution. Several tests to determine the best size of the list were made and was decided that is enough a Tabu List with size 6 for the three considered loads. Table 3 shows the attributes of the solutions that were maintained on the Tabu List according to the used neighborhood.

Table 3. Attributes of solutions maintained on the Tabu List

Neighborhood	Attribute 1	Attribute 2
1	Function Block	Function Block
2	Function Block	Processor
3	Message	Processor

The Simulated Annealing algorithm also needed a refined adjustment, before the collection of data. This adjustment is more complex, because it involves the initial temperature and the rate of reduction of the temperature. At first, a slow reduction of the temperature is desirable, but because the executions were made with limited time, it was necessary to adjust the algorithm so that it can take advantage of the available time in the best possible way. Among the three studied metaheuristics, Simulated Annealing is in which more parameters need to be configured. Such parameters are: initial temperature, final temperature, rate of reduction of the temperature (alpha) and the number of iterations per temperature. These parameters were configured considering the computational load and the execution time of the metaheuristic.

The experiments were made with a computer using a Pentium 1.7Ghz processor and 256Mbytes of main memory. The algorithms were implemented in C++. A total of 24 simple strategies and 100 medium and complex strategies were randomly generated. Each algorithm executed in two times, for 1 minute and for 3 minutes, and it supplied the best solution than it had found. The parameters of the three algorithm were empirically adjusted looking for the best behavior in executions of 1 and 3 minutes.

Strategies formed by loops with equal periods and with different periods were considered. In all the experiments that were made the Tabu Search was better than the other two evaluated metaheuristics. Figures 6 and 7 summarize the Tabu Search results considering the different loads, periods and execution time of the metaheuristic. All the satisfactory solutions supplied by the Tabu Search were found using neighborhood 3.

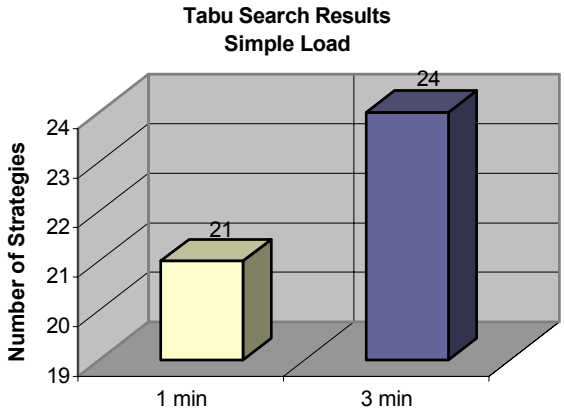

Figure 6. Tabu Search results for the simple load (P = 1000ms).

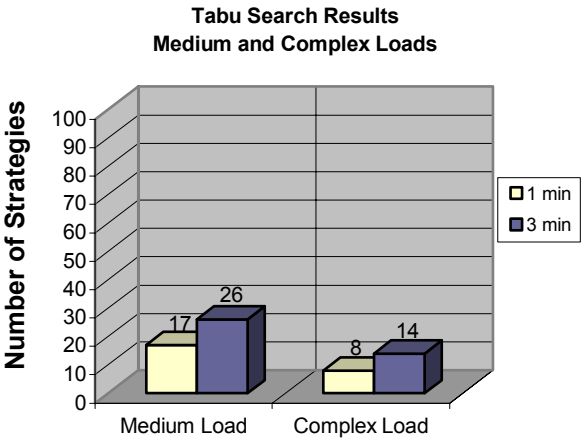

Figure 7. Tabu Search results for the medium and complex loads (P = 2000ms and P = 3000ms).

In the tests with different periods, the Tabu Search found 16 satisfactory solutions out of the 24 simple strategies, executed during one minute, and 18 satisfactory solutions with 3 minutes of execution of the metaheuristic, these two cases using the neighborhood 3. Fifty strategies were selected with different periods for each one of the medium and complex loads, for which none of the metaheuristics found the satisfactory solution (with 1 and 3 minutes of execution and using the two neighborhoods), but the Tabu Search was near in several cases to a satisfactory solution for the strategy.

The time of execution was extended for Simulated Annealing and Multi-Start to verify if they could find a satisfactory solution for some strategies. The used extended times were of 10 and 30 minutes. Although the time was extended, none of the two metaheuristics could find the satisfactory solution for the set of strategies, but the supplied solutions were better than the best found with 1 and 3 minutes of execution.

7. Conclusions

In this paper we presented experiments with the application of the Multi-Start, Simulated Annealing and Tabu Search metaheuristics for the solution of the allocation and scheduling problem of function blocks and periodic messages on the Foundation Fieldbus system. Representative computational loads from typical applications of the Foundation Fieldbus were used. The objective of the experiments wasn't to make an absolute performance analysis of the metaheuristic, but to compare them to each other, under controlled conditions.

The first verification was that Simulated Annealing requires a much bigger effort of adjustment than the other two metaheuristics, which are easier to configure. Simulated Annealing needed to pass for a refined adjustment, before the data collection could start.

In the experiments the Tabu Search presented better results than the other two methods. For the conditions used in the experiments, the Tabu Search was consistently better than the other two metaheuristics. For strategies formed by loops with equal periods, it was capable of finding satisfactory solutions, while the other ones don't. When increasing the execution time from 1 minute to 3 minutes, the Tabu Search was capable of increasing the number of strategies satisfactorily solved. Experiments where the loops had different periods were more difficult to solve, nevertheless the results of the Tabu Search were better. With regard to the neighborhoods, all the satisfactory solutions supplied by the Tabu Search were found using the Neighborhood 3.

The study stablished the superiority of the Tabu Search over the others studied methods, when considering problem instances typical of Foundation Fieldbus. A total of 224 scenarios composed by loads typical of this type of network were used, of real systems. The choice of the values used for the computation times of the blocks and the transmission time of the messages was made by a compilation based on catalogs of manufacturers.

It doesn't exist in the literature an indication as strong as the presented here in favor of the Tabu Search for the Foundation Fieldbus scheduling.

Finally, it is necessary to exercise caution about the results of the experiments. Firstly, the conclusions are valid just for computational loads and networks of the type analyzed here. The used scenarios are typical of Foundation Fieldbus, but they cannot be generalized for other communication networks. Another important subject is the difficulty to configure the Simulated Annealing. It is not possible to affirm that, with other configuration, Simulated Annealing would not be better. However, the empiric method of configuration used here didn't bring good results.

More details about the achieved experiments, inclusive with additional results, can be found in [11]. As a future work, the authors intend to consider other neighborhood definitions and new configurations for Simulated Annealing. Besides, the authors plan to investigate if more complex algorithms will be able to have the same effectiveness of Tabu Search. Also, to evaluate the impact of the execution time of the Tabu Search on the quality of its results.

Acknowledgment

This work is partially supported by a research grant from CNPq - The Brazilian National Council for Scientific and Technological Development.

References

[1] F. Lian, J. R. Moyne e D.M. Tibury, "Performance Evaluation of Control Networks: Ethernet, ControlNet and DeviceNet", IEEE Control Systems Magazine, USA, February, 2001, Pag. 1-2.

[2] W. Zhang, M.S. Branicky e S.M. Phillips, "Stability of Networked Control Systems", IEEE Control Systems Magazine, USA, 2001, Pag. 1-2.

[3] Fieldbus Foundation, "Technical Overview", Austin Texas, 2003. Available: http://www.fieldbus.org

[4] J. Berge, *Fieldbuses for Process Control: Engineering, Operation, and Maintenance*, ISA – The Intrumentation, Systems, and Automation Society, USA, 2002.

[5] S. Kirkpatrick, C. D. Gelatt e M. P. Vecchi, "Optimization by Simulated Annealing", Science, Vol. 220, Number 4598, May, 1983.

[6] P. E. Black, "NP-Complete", National Institute of Standards and Technology, September, 2004.

[7] R. Martí, *Multi-Start Methods*, Handbook of MetaHeuristics, Glover and Kochenberger Eds., 355-368, 2000.

[8] E. A. Silver, "An Overview of Heuristic Solution Methods", Working Paper University of Calgary, 2002.

[9] Smar, "Function Blocks Instruction Manual", 2002.

[10] K. Tindell, A. Burns, e A. Wellings, "Allocating Hard Real Time Tasks (An NP-Hard Problem Made Easy)", 1992.

[11] L. M. Franco V., Study About the Use of Meta-Heuristics for the Allocation and Scheduling of Function Blocks in Foundation Fieldbus Networks. Dissertation for Master Degree, Federal University of Santa Catarina, Brazil, March, 2005.