Improvement of Lagrangian Relaxation Convergence for Production Scheduling

Roman Buil, Miquel Àngel Piera, and Peter B. Luh

Abstract—**It is widely accepted that new production scheduling tools are playing a key role in flexible manufacturing systems to improve their performance by avoiding idleness machines while minimizing set-up times penalties, reducing penalties for do not delivering orders on time, etc. Since manufacturing scheduling problems are NP-hard, there is a need of improving scheduling methodologies to get good solutions within low CPU time. Lagrangian Relaxation (LR) is known for handling large-scale separable problems, however, the convergence to the optimal solution can be slow. LR needs customized parametrization, depending on the scheduling problem, usually made by an expert user. It would be interesting the use of LR without being and expertise, i.e., without difficult parameters tuning. This paper presents innovative approaches on the LR method to be able to develop a tool capable of solve scheduling problems applying the LR method without requiring a deep expertise on it. First approach is the improvement of an already existing method which use Constraint Programming (CP) to obtain better primal cost convergence. Second approach is called Extended Subgradient Information (ESGI) and it speed up the dual cost convergence. Finally, a set of step size rules for the Subgradient (SG) method are compared to choose the most appropriate rule depending on the scheduling problem. Test results demonstrate that the application of CP and ESGI approaches, together with LR and the selected step size rule depending on the problem, generates better solutions than the LR method by itself.**

Note to Practitioners—**Production scheduling tools are one of the keys in flexible manufacturing systems to improve its performance. These tools are usually based on optimization methods, as could be the Lagrangian Relaxation. The problems of using optimization methods are the need of time to get the solution, and the need of a high-specialized user to tune them. Therefore, optimization methods must be improved to use less time to obtain solutions and to do not need high-specialized users. This paper was motivated by these needs: reducing the CPU time when scheduling operations in production planning to permit quick replies to real-time perturbations into production processes; and making easier the use of production scheduling tools. This paper suggests new approaches for the Lagrangian Relaxation (LR) method applying Constraint Logic Programming (CLP) and improving the multipliers calculation (inside the Subgradient method) during the iterations to speed up the convergence of the LR method and make it easily tuned. Thus, the CPU time to find a solution is reduced and the results show that the use of the approaches reduces the needed knowledge (about the LR method application) to correctly tune the parameters to obtain good solutions. Therefore, an industry would be able to react to perturbations in less time and without a high-specialized user.**

Index Terms—**Constraint programming, Lagrangian relaxation (LR), production planning, scheduling.**

I. Introduction

THE increasing demand for on-time delivery of products and low production cost is forcing manufacturers to seek effective schedules. High-quality schedules are important to manufacturers in today's time-to-market competition because they can improve delivery performances and they can reduce inventory costs. However, obtaining a high-quality schedule within an acceptable CPU time is difficult because most manufacturing scheduling problems are NP-hard. Given a set of jobs with different number of operations each one and given a set of finite production resources, a manufacturing scheduling problem consist on the assignment of these operations to production resources under certain constraints and optimizing some objective function.

Lagrangian Relaxation (LR) was developed to solve manufacturing scheduling problems in [1] for a carefully established separable formulation. The LR approach is an iterative method which is used to divide a NP-hard problem into non-NP-hard subproblems by relaxing the coupling constraints through the Lagrange multipliers. These subproblems can be efficiently solved (which is done each iteration), and from their solutions, heuristics can be used to find a feasible schedule of the original problem. This process of finding a feasible solution is done every few iterations to obtain a feasible cost for the original problem. The best feasible cost is used to estimate the optimal dual cost, and it is also used to determine the step sizes to maximize the dual function. At the end of such multiplier updating iterations, the heuristic is applied again and the best feasible solution is chosen as the solution to the original problem [1]. Unfortunately, this method do not always provides good feasible solutions because LR convergence is very sensitive to different parameters and highly dependent on problem characteristics [2]. In the authors opinion, this fact constrained the widely use of LR as optimization method for scheduling problems because the results depend on how the parameters are tuned.

Manuscript received February 12, 2011; revised July 20, 2011; accepted September 03, 2011. Date of publication September 26, 2011; date of current version December 29, 2011. This paper was recommended for publication by Associate Editor C. Chu and Editor Y. Narahari upon evaluation of the reviewers' comments. This work was supported in part by the CICYT Spanish Program TRA2008-05266/TAIR and in part by the AGAUR (Generalitat de Catalunya) Program 2009 SGR 629.

R. Buil and M. A. Piera are with the Department of Telecommunications and Systems Engineering, Universitat Autònoma de Barcelona, Bellaterra 08193, Barcelona, Spain (e-mail: roman.buil@uab.cat; miquelangel.piera@uab.cat).

P. B. Luh is with the Department of Electrical and Computer Engineering, University of Connecticut, Storrs Mansfield, CT 06269-2157 USA (e-mail: peter.luh@uconn.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2011.2168817

Constraint Programming (CP) is a paradigm tailored to hard search problems [3], such as manufacturing scheduling problems, and can be used when the problem is written as a Constraint Satisfaction Problem (CSP). A CSP consist of a set of variables, a domain for each variable specifying the values to which the variable may be assigned, and a set of constraints on the variables, restricting the values that they can simultaneously take [4]. This set of domains is the search space for the solution. This space can be reduced either when a variable is instantiated or when the domain of a variable is cut. CP guarantee solution feasibility due to the satisfaction of constraints. Although a manufacturing scheduling problem can be defined as a CSP, since CP may not be able to take advantage of the problem structure to decompose it into subproblems. Consequently, the search space could become too large and to find the optimal solution may require nonfeasible CPU time.

A method combining LR and CP was presented in [5], and its results are better than results of the LR method by itself. This method applies CP to find feasible solutions from the unfeasible ones, at some iterations. Thus, it improves the convergence of the primal cost if the parameter which determines the number of iterations between each CP application is properly tuned. After the analysis of the LR and CP method results, authors concluded that better performance on convergence can be obtained if dual cost is increased at the same time that primal cost is decreased, and this is the reason of a new modification on the LR method.

This paper presents a new approach called Extended Subgradient Information (ESGI) which tries to improve the dual cost at the iterations where dual cost decrease from the previous one. If dual cost can be improved by obtaining a new solution with dual cost greater than the dual cost of the solution at iteration n, this solution updates the solution of iteration $n+1$, and the method continues from it.

In addition to this approach, the CP approach is applied using new constraint structure which ensures a better performance when finding feasible solutions. Thus, the search within a CP iteration is less time consuming than the search using previous constraint structure.

Although CP and ESGI approaches contribute to obtain better solutions using LR, there is still the need of finding the appropriate step size rule for each type of scheduling problem. Therefore, this paper also presents different step size rules for the multipliers updating process. Before applying CP and ESGI, the LR method is used to test selected step size rules with minimal tuning because the idea is not to get the best possible solution of tested problems, but to improve the convergence of the LR method using the best step size rule, CP and ESGI. Usually, the rule with best performance will be chosen; however, different rules are selected in test results of this paper to show the possible impact of CP and ESGI on the solutions quality.

This paper is organized as follows. Section II presents the problem formulation, which is needed to explain the methods. The LR method is presented in Section III, and Section IV describes the LR and CP method. Section V presents the ESGI which improves the dual cost convergence, and Section VI introduce the tested step size rules. Test results are presented in Section VII and, finally, Section VIII presents the conclusions and future work.

II. Problem Formulation

As mentioned before, LR is known for handling separable problems. The particular case used for this paper is a job shop scheduling problem which is structurally separable.

Following the formulation presented in [1] and [5], the problem is to schedule N parts on H types of machines with time horizon K to minimize a weighted tardiness and earliness criterion [6]. Each machine type h, $(h = 1, 2, \ldots, H)$ has M_{kh} available machines at time k, $k = 1, 2, \ldots, K$, and J_i operations are required for the completion of each part i $(i = 1, 2, \ldots, N)$. Operation j of part i is denoted as (i, j). Each operation (i, j) is performed on a machine of a given machine type, and this machine type selected to process operation (i, j) is denoted as h_{ij}, and the set of machine types capable to perform operation (i, j) is denoted by H_{ij}. The decision variables of the scheduling problem are the beginning times of all operations $\{b_{ij}\}$ and the machine types selected to perform each operation $\{h_{ij}\}$. This discrete-time, integer programming formulation and the solution methodology developed in this paper are applicable even if some parts are not available at time 0, even if setup times are considered, and even if stock rupture or other alterations are introduced. However, for simplicity, setup time, stock rupture or other alterations are not considered in this formulation. The objective functions and constraints are presented next.

Objective Function: The scheduling goal of on-time delivery for individual parts is modeled as penalties on part delivery tardiness $T_i = \max[0, c_i - d_i]$ (where c_i is the completion time of part i and d_i is the due date of part i); and as penalties on releasing orders too early, $E_i = \max[0, db_i - b_i]$ (where $db_i = d_i + 1 - \sum_j t^*_{ijh}$ is the desired beginning time, b_i is the beginning time of part i, and t^*_{ijh} is the minimum processing time of operation (i, j) in machine type $h \in H_{ij}$). The objective function of the problem is

$$J = \sum_i \left[\omega_i T_i^2 + \beta_i E_i\right] \tag{1}$$

where ω_i and β_i are weights associated with tardiness and earliness penalties. The square on tardiness reflects that a part becomes more critical with each time unit passing its due date. The problem is to minimize (1) subject to the following constraints and requirements.

Parts Availability: Each part is available at a certain unit time k_{i0}, $k_{i0} = 1, 2, \ldots, K$. This set of constraints is not used if all parts are assumed to be available at time 0

$$b_{i0} \geq k_{i0}, \quad i = 1, \ldots, N. \tag{2}$$

Processing Time Requirements: For each part, the processing of each operation requires a machine of a specific type for some prespecified units of time, and must satisfy the following processing time requirements:

$$c_{ij} = b_{ij} + t_{ijh} - 1, \quad i = 1, \ldots, N; \quad j = 1, \ldots, J_i \tag{3}$$

where c_{ij}, b_{ij}, and t_{ijh} represent, respectively, the completion time, the beginning time and the processing time of (i, j). The h in t_{ijh} represents the machine type where the operation is processed.

Operation Precedence Constraints: Each operation may be started only after the completion of its preceding operation

$$c_{ij} + 1 \leq b_{ij+1}, \quad i = 1, \ldots, N; \quad j = 1, \ldots, J_i - 1 \tag{4}$$

where b_{ij+1} represents the beginning time of $(i, j+1)$.

Machine Capacity Constraints: There is a number of possible available machines for each machine type and they determine the capacity for each machine type. The number of operations assigned to machine type h at time k should be less than or equal to M_{kh}, the number of machines available at that time

$$\sum_{ij} \delta_{ijkh} \leq M_{kh}, \quad k = 1, \ldots, K; \quad h = 1, \ldots, H \tag{5}$$

where δ_{ijkh} is a 0–1 variable and equals 1 if (i, j) is being processed by a machine of type h at time k; it equals 0 otherwise. The values of these variables, δ_{ijkh}, are determined once the beginning times b_{ij} of all operations are decided.

The problem is to minimize (1) through selecting appropriate machine types and beginning times, subject to constraints (2)–(5). Note that parts availability, precedence constraints and processing time requirements relate to individual jobs, and the objective function is also jobwise additive. Only capacity constraints couple across jobs and make the problem intractable. This formulation is thus separable.

III. Lagrangian Relaxation (LR) Method

The complexity of the scheduling problem motivates the use of decomposition techniques. LR has been used by Luh and his research team in [1], [5]–[7] and in other publications to achieve good decomposition of scheduling problems relaxing one or more sets of constraints. The formulation presented in Section II can be decomposed into partwise subproblems by just relaxing the coupling machine capacity constraints. The following subsections present the different parts of the LR method.

A. Problem Decomposition

As it has been presented in the LR framework [6], the machine capacity constraints (5) are relaxed by using non-negative Lagrange multipliers π_{kh}, and the relaxed problem is obtained as

$$\min_{b_{ij}} L, \quad \text{with}$$

$$L \equiv \sum_i \left[\omega_i T_i^2 + \beta_i E_i + \sum_{kh} \pi_{kh} \left(\sum_{ij} \delta_{ijkh} - M_{kh} \right) \right] \tag{6}$$

subject to (2)–(4).

Since the formulation is separable, for the given set of multipliers it is possible to extract a minimization subproblem for each part

$$\min_{b_{ij}, h_{ij}} L_i, \quad \text{with}$$

$$L_i \equiv \omega_i T_i^2 + \beta_i E_i + \sum_j \left[\sum_{k=b_{ij}}^{c_{ij}} \pi_{kh_{ij}} \right] \tag{7}$$

subject to (2)–(4).

Note the fact that $\delta_{ijkh} = 0, \forall h \neq h_{ij}$ is used and h_{ij} is the machine type selected to process operation (i, j).

Each subproblem is to schedule the operations of a single part to minimize its tardiness and earliness penalties and the costs for using machines. These subproblems can be effectively solved by using Dynamic Programming (DP) (see Section III-B). Let L_i^* denote the minimum subproblem cost of part i with given multipliers, the high-level Dual problem is then given by

$$\max_{\pi_{kh}} q, \quad \text{with}$$

$$q \equiv -\sum_{kh} \pi_{kh} M_{kh} + \sum_i L_i^* \tag{8}$$

where $\pi_{kh} \geq 0, \forall h, \forall k$. Let q^* denote the optimal dual cost.

B. Solving Subproblems

It has been shown in [6] and [8] that each part subproblem is a multistage optimization problem, and can be efficiently solved by using DP with pseudo-polynomial complexity. In this paper, the Backward Dynamic Programming (BDP) algorithm is used.

The BDP algorithm takes the beginning times and machine types selected for operations as decision variables and starts with the last stage of a job. The cumulative costs are obtained recursively and subject to operation precedence constraints (4), processing time requirement constraints (3), and parts availability constraints (2). The algorithm compute the cost of the last operation (i, J_i) for all possible b_{i,J_i} and h_{i,J_i}

$$V_{i,J_i}(b_{i,J_i}, h_{i,J_i}) = \omega_i T_i^2 + \sum_{k=b_{i,J_i}}^{c_{i,J_i}} \pi_{kh_{i,J_i}} \tag{9}$$

then, moves backward until the first operation. Operations (i, j) with $j = 1 \ldots J_i - 1$ are obtained recursively solving

$$V_{ij}(b_{ij}, h_{ij}) = \min \left\{ \sum_{k=b_{ij}}^{c_{ij}} \pi_{kh_{ij}} + V_{i,j+1}(b_{i,j+1}, h_{i,j+1}) \right\} \tag{10}$$

and operation $(i, 1)$ is obtained solving

$$V_{i1}(b_{i1}, h_{i1}) = \min \left\{ \beta_i E_i + \sum_{k=b_{i1}}^{c_{i1}} \pi_{kh_{i1}} + V_{i2}(b_{i2}, h_{i2}) \right\}. \tag{11}$$

The function $V_{ij}(b_{ij}, h_{ij})$ is the cumulative cost for all operations including and succeeding (i, j), and $(\omega_i T_i^2 + \sum_{k=b_{i,J_i}}^{c_{i,J_i}} \pi_{kh_{i,J_i}})$, $\sum_{k=b_{ij}}^{c_{ij}} \pi_{kh_{ij}}$, and $(\beta_i E_i + \sum_{k=b_{i1}}^{c_{i1}} \pi_{kh_{i1}})$ for $j = 1$, are the stagewise costs. The optimal subproblem cost L_i^* is then obtained as the minimal cumulative cost at the first stage. Finally, optimal beginning times and machine types selected for operations can be obtained by tracing forwards the stages.

Multipliers must be fixed before solving subproblems, therefore, they are updated each iteration (see Section III-C) before BDP. Multipliers, beginning times and machines assignation form a solution of the dual problem.

C. Solving the Dual Problem

To solve the dual problem related to (8), the subgradient (SG) method is used [9], [10].

Given an initial set of multipliers π_{kh}^0, the subproblems are solved individually and beginning times b_{ij}^0 at the first iteration are obtained. At the $n+1$ iteration, the multipliers are updated according to

$$\pi_{kh}^{n+1} = \pi_{kh}^n + s^n g^n \tag{12}$$

where s^n is the step size at the nth iteration, and g^n is the subgradient of q at the nth iteration. The subgradient component of machine type h at time k is given by the capacity constraints as

$$g^n = g\left(b_{ij}^n\right) = \sum_{ij} \delta_{ijkh}^n - M_{kh} \tag{13}$$

where b_{ij}^n are the beginning times at iteration n. The step size s^n is determined by

$$s^n = \frac{\alpha_n(q^* - q^n)}{\|g^n\|^2}, \quad 0 < \alpha_n < 2 \tag{14}$$

where q^* is the optimal solution of (8), q^n is the dual cost obtained at iteration n, and α_n is a parameter which can change over iterations.

In reality, q^* is not known. Therefore, an estimated $\hat{q}^*$ is used. This $\hat{q}^*$ is taken as an upper bound of the optimal dual cost q^*, and is obtained by constructing a feasible schedule applying a simple heuristic.

It can be said that step size consists in two parameters: One depends on the evolution of the problem, which is $(q^* - q^n)/\|g^n\|^2$, and the other one depends on a previous fixed rule, and it is α_n. Section VI present some step size rules used as α_n for the testing results.

D. Constructing Feasible Solutions

The solution of the dual problem is generally an infeasible solution of the scheduling problem (primal problem) because some capacity constraints (5) might be violated since they are relaxed. Otherwise, parts availability constraints (2), processing time requirements (3), and operation precedence constraints (4) are always satisfied in view of the way subproblems (7) are solved. To construct a feasible solution, the algorithm presented in [1] is applied where a list is created by arranging all the operations in the order of the beginning times of dual solution. Operations are then scheduled according to the list as required machines become available. A greedy heuristic based on the incremental change in the cost function J (1) is used to decide which operation(s) should be delayed when some capacity constraint is violated. If these delays cause precedence constraints violations, the subsequent operations of the delayed ones violating precedence constraints are also delayed by one time unit.

E. Iterative Method

As mentioned before, LR is an iterative method. At each iteration, multipliers are updated, subproblems are optimally solved to obtain dual solutions, and primal cost is updated by heuristics as appropriate.

An iterative method always needs a stop criteria. The presented method stops either when step size becomes too small, when the distance between q^n and q^* becomes too small, or when a fixed number of iterations is reached. The equation to evaluate these parameters are defined as follows:

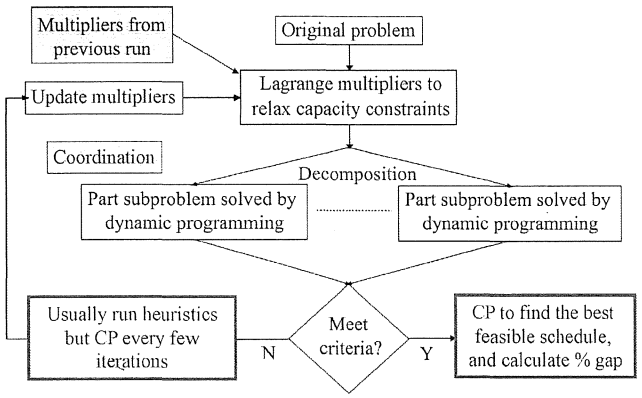

Fig. 1. Schema of the LR and CP method.

$$\left\|\pi_{kh}^{n+1} - \pi_{kh}^n\right\| \leq \xi \tag{15}$$

$$Maximum\ number\ of\ iterations = n \tag{16}$$

$$GAP(\%) = \frac{(\hat{q}^* - q^n)}{q^n} * 100 \leq Gap\ Limit \tag{17}$$

where GAP denote the relative distance between $\hat{q}^*$ and q^n. n, $Gap\ limit$ and ξ are fixed depending on the problem.

IV. LR and CP Method

As presented in [5] CP is used during the iterative LR method to find feasible solution using Time Windows (TWs) constructed around the unfeasible solutions. At the end of the iterations CP is also used but with different TW. Fig. 1 presents the schema of the method.

A. Time Windows Construction

As presented in Section III-B, for a given set of multipliers, subproblems are optimally solved one by one, and these subproblems solutions usually are an unfeasible solution for the original problem. TW $[b_{ij_{low}}^n, b_{ij_{high}}^n]$ around subproblems solutions are constructed, at the nth iteration, to control the rescheduling of operations. The TW bounds are driven so that the cumulative costs of the operation at the lower bound $(b_{ij_{low}}^n)$ and at the upper bound $(b_{ij_{high}}^n)$ are within ε-neighborhood of the optimal cumulative cost, i.e.,

$$\begin{aligned} b_{ij_{low}}^n = \min \{k | k \leq b_{ij}^n \ and\ V_{i,j}^n(k', h_{ij}) \\ \leq (1+\varepsilon^n) V_{i,j}^n\left(b_{ij}^n, h_{ij}\right)\ for\ any\ k', \\ k \leq k' \leq b_{ij}^n\}, \\ b_{ij_{high}}^n = \max \{k | k \geq b_{ij}^n \ and\ V_{i,j}^n(k', h_{ij}) \\ \leq (1+\varepsilon^n) V_{i,j}^n\left(b_{ij}^n, h_{ij}\right)\ for\ any\ k', \\ k \leq b_{ij}^n \leq k' \leq k\} \end{aligned} \tag{18}$$

where b_{ij}^n is the beginning time of operation j of job i at iteration n; $V_{ij}^n(k, h_{ij})$ is the cumulative cost of operation j of part i performed in a machine of type h_{ij} at time slot k, at iteration n;

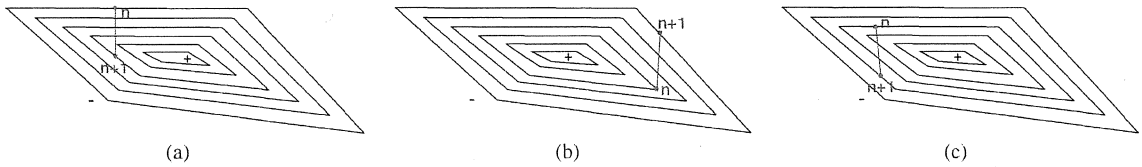

Fig. 2. Contour lines of dual cost and possible situations for consecutive iterations.

and $\varepsilon^n > 0$ is a parameter which value is determined depending on the previous use of CP

$$\varepsilon^{n+1} = \begin{cases} \varepsilon^n \rho, & \text{if previous use of CP finds a solution} \\ \varepsilon^n / \rho, & \text{if previous use of CP does not find a solution} \end{cases} \tag{19}$$

where n is the iteration number and ρ is a parameter such that $0 < \rho < 1$.

Values for ε^0 and ρ by doing some preliminary experiments. ε^0 use to be as small as possible taking into account that should be large enough to define time windows containing some solution. However, it should also made time windows small enough to do not be too large, because large time windows implies large search space which difficult the finding of solution. $\rho = 0,9$ or $\rho = 0,95$ is usually fixed to resize time windows between two successive uses of CP. Thus, there are not large variations in the size of these time windows.

Besides using this ε, minimum and maximum length for TW can be fixed if previous experiments have given this knowledge.

B. Constraint Programming

A CSP consist in a set of variables with their domains and with constraints on them. A CSP must be defined each time CP is used in the LR and CP method, and the one defined for the problem presented in Section II is the following.

- Decision variables: $b_{ij}^n, h_{ij}^n, \forall i, \forall j$.
- Domains: $b_{ij}^n :: 0 \ldots K, h_{ij}^n \in H_{ij}, \forall i, \forall j$, where $b_{ij} :: 0 \ldots K$ means that b_{ij}^n can take values from 0 to K (the largest time unit).
- Constraints: part availability constraints (2), processing requirements constraints (3), precedence constrains (4), capacity constraints (5), and the following time window constraints based on (18)

$$b_{low_{ij}}^n \leq b_{ij}^n, \quad \forall i, \quad \forall j, \tag{20}$$

$$b_{ij}^n \leq b_{up_{ij}}^n, \quad \forall i, \quad \forall j \tag{21}$$

where n is the iteration.

A CSP can also include a cost function and a domain bounding this cost. For this problem, the cost function is J, defined in (1), and its domain is

$$q^n \leq J^n < \hat{q}^*. \tag{22}$$

If this J^n exists, $\hat{q}^*$ is updated, otherwise, $\hat{q}^*$ will not change. When iterations stop, the last unfeasible schedule generated by LR is used to construct the final TW, which are quite larger than the TW used during the iterations. J^n is minimized to find the best solution within these TW instead of to search just a feasible solution as is done during the iterations.

A Constraint Logic Programming (CLP) software system called ECL^{i}PS^e is used to perform the CP approach. ECL^{i}PS^e is largely backward-compatible with Prolog and supports different programming languages. It also provides several libraries of constraint solvers which can be used in application programs. One of these libraries permit the implementation of the capacity constraints (5) presented in this paper. Therefore, the implementation of the capacity constraints made by the authors of this paper in [5] is replaced by this new implementation developed by the ECL^{i}PS^e development team. Robustness and propagation of value assignments are improved and it becomes in better convergence of the method.

The CP search could be too much time consuming, therefore, a maximum CPU time is fixed for that search to do not overly increase the total CPU time. Due to the definition of the CSP, the finding of a solution implies a improve of the primal cost which means better convergence and less CPU time.

The ideal situation is to be able to fix the number of iterations between each CP use for each type of problem, or to use a variable number depending on some rules. However, these results (see Section VII) include comparative tables with different amounts of iterations. Test with best performance fixes this number of iterations for test applying CP and ESGI together with the LR method.

V. Extended Subgradient Information

The LR and CP method converges faster than the LR method because primal cost is decreased faster by using CP. If the increasing of the dual cost could be also speeded up, then the convergence of the method would be even better.

Analyzing the evolution of the dual cost during iterations, it is observed that dual cost do not increases its value every iteration. However, since the dual function is piecewise linear concave, in some situations it is possible to find better dual solutions on the SG direction, and this is the objective of ESGI approach. Fig. 2 shows the three different possible situations on the dual cost contour lines. n and $n+1$ denote the point of the dual solution at iteration n and at iteration $n+1$, respectively. The possible situations and their consequences are the following.

- The best situation is to get greater dual cost at iteration $n+1$ than at iteration n, as Fig. 1(a) shows. In this case, LR continues.
- In Fig. 2(b), the dual cost of iteration $n+1$ is less than at iteration n; and there is no point on the SG direction with dual cost greater than the dual cost at iteration n. In this case, LR also continues.
- Finally, Fig. 2(c) present the case where a new point solution with greater dual cost than the dual cost at iteration n and $n+1$ can be found. This is the case when dual solution of iteration $n+1$ is updated by this new solution with better cost.

The idea of the ESGI is to find the best dual solution on the SG direction and between the point of the iteration n and the point of iteration $n+1$ (interval called segment); however, this search would be too much CPU time consuming because all the subproblems should be solved for each possible point on the SG direction. The current ESGI just evaluates one point and if the dual cost of the solution in this new point is greater than the dual cost at iteration n, the new solution substitutes the solution of iteration $n+1$ and the LR method continues from this new solution (set of multipliers, beginning times, and machines assignation). Otherwise, if the dual cost of the solution in this new point is less than the dual cost at iteration n and $n+1$, the LR method continues from the solution at iteration $n+1$.

Current testing results (Section VII) check one point on the segment, and this point is calculated using a new parameter $(0 < \sigma < 1)$ which multiplies the step size s^n. This σ takes just one fixed value during all the iterative method, which is taken from 0.1 to 0.9 using 0.1 intervals. The comparative results with different values for σ permit to select the value with best performance. The selected σ is used for the final test which applies LR, CP, and ESGI.

VI. Step Size Rules

As mentioned in Section III-C, different rules are used to determine the value of the parameter α_n of (14). Reference [10] presents different step size rules which they use as step size s^n in (14). In this paper, the value of $(q^* - q^n)/\|g^n\|^2$ is added to the step size rules defined in [10]; therefore, α_n takes these rules. Additionally, other rules are also tested taking into account that α_n must satisfy $0 < \alpha_n < 2$. All the used rules are presented below and n indicates the number of the iteration.

a) $\alpha_n = h^n$, where h is a constant close to 2.

b) $\alpha_n = h^{1+\lfloor n/p \rfloor}$, where h is a constant close to 2, and p is a constant, which indicates the number of iterations between each decrease of α_n.

c) $\alpha_n = \alpha_{n-1} * h$, if dual has not been improved in iteration $n-1$,
$\alpha_n = \alpha_{n-1} * ((3-h)/2)$, if dual has been improved in iteration $n-1$,
where $h < 1$, $\alpha_0 = h$ and $\alpha_n = 2 * h$ if $\alpha_n \geq 2$ is obtained.

d) $\alpha_n = \alpha_{n-1} * h$, if dual has not been improved in iteration $n-1$,
$\alpha_n = \alpha_{n-1}/h$, if dual has been improved in iteration $n-1$,
where $h < 1$, $\alpha_0 = 2 * h$ and $\alpha_n = 2 * h$ if $\alpha_n \geq 2$ is obtained.

e) $\alpha_n = \alpha_{n-1} * h$, if dual has not been improved in iteration $n-1$,
$\alpha_n = \alpha_{n-1}$, if dual has been improved in iteration $n-1$,
where $h < 1$, $\alpha_0 = 2 * h$ and $\alpha_n = 2 * h$ if $\alpha_n \geq 2$ is obtained.

f) $\alpha_n = h$, where h is a constant.

g) $\alpha_n = h/\|g^n\|^2$, where h is constant and g^n is the subgradient introduced in Section III-C.

h) $\alpha_n = a/(b+n)$, where $a > 0$, $b \geq 0$, and α_n satisfies

$$\sum_{n=1}^{\infty} \alpha_n^2 < \infty, \quad \sum_{n=1}^{\infty} \alpha_n = \infty.$$

i) $\alpha_n = a/\sqrt{n}$, where $a > 0$ and α_n satisfies

$$\lim_{n\to\infty} \alpha_n^2 = 0, \quad \sum_{n=1}^{\infty} \alpha_n = \infty.$$

The idea is to select one of these rules for a type of scheduling problem and use it when the LR method is applied together with CP and ESGI. In this paper, all the rules are compared solving the problems applying the LR method. Some of these rules are selected later on to be used for the rest of the tests made at Section VII.

VII. Test Results

The LR method and the approaches have been implemented by using the objective programming language C++ and ECL^{i}PS^e. The main program is developed by using C++, and when CP is needed, C++ establish the communication with ECL^{i}PS^e, and the Prolog algorithm for the problem is executed. The results are saved into C++ variables and used by the C++ main program. Presented testing have been performed on a Intel core 2 Duo, 2.26 GHz Macbook with 4 GB of RAM and Mac OS X System. No advanced search strategies have been used for the CP approach; even so, a simple backtracking is enough for these tests after the constraints implementation improvement.

Three job shop scheduling problems have been used for these tests. A quite simple job shop scheduling problem with due dates presented in [5] is used for Example 1. This first example compares different results depending on the initial solutions obtained by using LR and all the implemented step size rules. Three different LR tests with different final GAP and number of iterations are selected to show the impact of the CP approach (Option 1), the ESGI approach (Option 2), and both together (Option 3) with LR, depending on the quality of the LR solution. Thus, selected tests are the following.

- The one with less number of iterations and reasonable GAP.
- One with reasonable GAP and whatever amount of iterations.
- One with GAP higher than the Gap Limit defined by (17) and more iterations than the other selected tests.

For Example 2, a more complete job shop scheduling problem with due dates taken from [1, Table IV] is used. In that case, two step size rules are selected.

- The one with less GAP.
- One reaching the maximum number of iterations and GAP greater than GAP Limit (to test if it is possible to get good solutions if the LR parameters tuning is not good at all).

The approaches are then compared using both rules, as made in Example 1.

Example 3 presents results of ten different instances randomly generated by the authors. Two tests are made for each instance, one using the LR method, and another one using Option 3 (LR, CP, and ESGI). Step size rules and CP and ESGI configurations are randomly fixed for each instances. LR parameters are fixed and equal for the ten instances.

TABLE I
RESULTS OF EXAMPLE 1 USING LR METHOD WITH DIFFERENT STEP SIZE RULES

Step Size	Num. of iter.	CPU Time (sec)	Primal Cost	GAP (%)
a	335	0	643	5.93
b.05	162	0	640	4.92
b.20	148	1	640	4.92
b.50	139	0	640	4.92
c	443	1	655	8.99
d	443	1	655	8.99
e	443	1	655	8.99
f	570	1	651	5.00
g	307	1	652	4.99
h	415	0	655	7.73
i	295	0	655	4.97

A. *Example 1*

The job shop schedule problem used for this example consists in two different types of machines with a capacity of three parallel resources for type 1 and a capacity of two parallel resources for type 2. There are 16 jobs with three operations each one, operations 1 and 2 must be processed in machine type 1 and operation 3 in machine type 2. Each operation has its process time and there are tardiness weights taken fixed values from 10 to 100 and earliness weights taken fixed values from 1 to 10. Due dates for each job are distributed along the time without overcome the time horizon, which is 50 time units. All parts are assumed to be available at time 0 for simplicity.

The parameters fixed for the stop criteria ((15)–(17)) are

$$\xi = 0.01, \quad n = 1000, \quad Gap\ Limit = 5\%.$$

The LR method is applied using different step size rules. All rules are applied once but rule (b) has been applied three times with different values of p ($p = 5$, 20, 50). Table I shows the results. Column "Step Size" indicates the step size rule, using $b.05, b.20, b.50$ for rule (b) with different values of p, and the corresponding index for the other rules (see Section VI). The rest of the columns are in order: number of used iterations, CPU time consumed, final primal cost, and final GAP. To comment the results, tests are named using the index of the applied step size rule.

It is easy to see that stop criteria of test a, c, d, e, and h is the proximity of the multipliers of consecutive iterations (15); and all the others tests have gotten a GAP less than the fixed GAP Limit (17). There is no test reaching the total number of permitted iterations (16).

The selected tests following the explanation in Section VII, and keeping the order are: $b.50, i$ and h. Therefore, Option 1...3 are applied to solve the problem using the step size rules $(b.50)$, (i) and (h). The results are compared to show the impact and benefits of the approaches and their combination.

1) Step Size Rule $(b.50)$*:* In Table II, the number of iterations decrease as CP is used most often, except when using CP every five iterations and at each iteration. Analyzing the obtained GAP's, there is an increase of GAP's inversely proportional to the decrease of iterations. This proportion appears because as faster decreases the primal cost (which converge to the optimum in all the test in Table II using CP) the more difficult it is to increase the dual cost, due to the use of these primal cost for the multipliers update during the iterations [see $\hat{q}^*$ as approximation of q^* in (14)]. Since CPU times are similar, but when using CP at every five iterations and every single iteration, it is necessary to make a balance between the number of iterations and GAPs to decide which is the best result when using CP (Option 1), and select it for its use in Option 3. The authors consider the minimum GAP with less number of iterations as the best result, if the CPU time is reasonable. Thus, CP every 30 iterations is selected.

TABLE II
RESULTS APPLYING LR AND CP METHOD (OPTION 1) USING STEP SIZE RULE $(b.50)$ ON EXAMPLE 1

Iter. CP	Num. of iter.	CPU Time (sec)	Primal Cost	GAP (%)
0	139	0	640	4.92
40	121	2	634	4.45
30	111	2	634	4.45
20	95	3	634	4.62
10	76	4	634	4.97
5	81	10	634	4.97
1	95	57	634	4.97

TABLE III
RESULTS APPLYING LR AND ESGI (OPTION 2) USING STEP SIZE RULE $(b.50)$ ON EXAMPLE 1

σ	Num. of iter.	CPU Time (sec)	Primal Cost	GAP (%)
0.1	135	0	640	4.92
0.2	255	0	655	4.97
0.3	254	0	655	4.97
0.4	249	1	655	4.97
0.5	243	0	655	4.97
0.6	242	1	655	4.97
0.7	244	0	655	4.97
0.8	239	0	655	4.97
0.9	235	0	655	4.97

TABLE IV
RESULTS APPLYING LR, CP AND ESGI (OPTION 3) USING STEP SIZE RULE $(b.50)$ ON EXAMPLE 1

Iter. CP	σ	Num. of iter.	CPU Time (sec)	Primal Cost	GAP (%)
30	0.1	107	2	634	4.45

Table III presents the results when applying Option 2, and they show that ESGI do not influence too much in the final results if just LR already gets good solutions in a small number of iterations when the problem is quite simple. A σ value has to be selected to used it in Option 3. In this case, $\sigma = 0.1$ is the test with the minimum number of iteration and it is selected.

Results of Option 3 presented in Table IV improve the results of Option 2 because a smaller GAP is gotten with less iterations. The obtained GAP is the same than in Option 1 using less iterations. Using Option 3, the optimum primal cost is obtained as using Option 1, thanks to the CP approach. However, less iteration are needed to get the same GAP due to the use of ESGI has increased the dual cost faster.

Previous results are obtained using the step size rule which gives the best performance. But what happens if the number of iterations to obtain a good solution (inside the GAP Limit) is greater? It could be because the problem cannot be solved using

TABLE V
RESULTS APPLYING LR AND CP METHOD (OPTION 1) USING STEP SIZE RULE (i) ON EXAMPLE 1

Iter. CP	Num. of iter.	CPU Time (sec)	Primal Cost	GAP (%)
0	295	0	655	4.97
40	156	2	634	2.42
30	137	2	634	2.59
20	114	3	634	2.76
10	91	4	634	3.09
5	56	4	634	3.43
1	77	46	634	4.97

TABLE VI
RESULTS APPLYING LR AND ESGI (OPTION 2) USING STEP SIZE RULE (i) ON EXAMPLE 1

σ	Num. of iter.	CPU Time (sec)	Primal Cost	GAP (%)
0.1	285	0	655	4.97
0.2	296	1	655	4.97
0.3	264	0	655	4.97
0.4	223	1	655	4.97
0.5	245	1	655	4.97
0.6	191	0	655	4.97
0.7	292	1	655	4.97
0.8	286	1	655	4.97
0.9	278	1	655	4.97

TABLE VII
RESULTS APPLYING LR, CP, AND ESGI (OPTION 3) USING STEP SIZE RULE (i) ON EXAMPLE 1

Iter. CP	σ	Num. of iter.	CPU Time (sec)	Primal Cost	GAP (%)
40	0.6	101	2	634	2.26

TABLE VIII
RESULTS APPLYING LR AND CP METHOD (OPTION 1) USING STEP SIZE RULE (h) ON EXAMPLE 1

Iter. CP	Num. of iter.	CPU Time (sec)	Primal Cost	GAP (%)
0	415	0	655	7.73
40	348	4	634	4.28
30	391	5	634	4.28
20	281	5	634	4.45
10	161	5	634	4.45
5	100	5	634	4.62
1	467	367	634	6.38

TABLE IX
RESULTS APPLYING LR AND ESGI (OPTION 2) USING STEP SIZE RULE (h) ON EXAMPLE 1

σ	Num. of iter.	CPU Time (sec)	Primal Cost	GAP (%)
0.1	389	0	655	7.20
0.2	369	1	655	7.38
0.3	373	0	655	7.55
0.4	374	0	655	7.20
0.5	375	0	655	7.55
0.6	395	0	655	7.73
0.7	443	0	655	7.73
0.8	377	0	655	7.55
0.9	377	1	655	7.55

TABLE X
RESULTS APPLYING LR, CP, AND ESGI (OPTION 3) USING STEP SIZE RULE (h) ON EXAMPLE 1

Iter. CP	σ	Num. of iter.	CPU Time (sec)	Primal Cost	GAP (%)
40	0.4	320	3	634	3.93

less iteration, then the situation would be the same as before with the step size rule $(b.50)$. However, imagine that the best LR solution is the one using step size rule (i). Which impact CP or/and ESGI could have on the results? Following tables show them.

2) Step Size Rule (i): As in previous results, the use of CP reduce the number of iterations as it can be seen in Table V. In this case, CP cause more impact than using step size rule $(b.50)$. GAPs are better in most of the cases and CPU times have the same behavior, being the use of CP at each iteration, the most CPU time consuming case. Selected CP configuration for Option 3 is to use CP every 40 iterations because it has the best GAP.

Table VI shows that in this case ESGI obtain similar results than in Table III. Primal costs and GAPs are the same for all the configurations, and there are variations in the number of iterations. The best result is obtained by $\sigma = 0.6$ and it use 191 iterations. This is the selected σ for Option 3.

The result of the last test presented in Table VII and using Option 3 improve all the other results with better GAP obtained with less iterations. The CPU time is the same as using CP every 40 iterations in Table V.

3) Step Size Rule (h): The last selected step size rule (h) simulate a case where number of iterations is high and GAP Limit is not reached. Results in Tables VIII and IX are the results applying Options 1 and 2, respectively. These results are similar than the other presented results with step size rules $(b.50)$ and (i). Table X shows the result of Option 3, and it improves the results of Options 1 and 2. Gap and number of iterations are improved.

Analyzing the results of Example 1, it can be seen that the LR method is not robust enough because it depends on different parameters. One of them is the step size parameters. Notice that results of the LR method present variations depending on the used step size rule. However, results obtained with LR, CP, and ESGI with the three step size rule selected are quite similar. Therefore, it is possible to ensure that applying the CP and ESGI approaches the results do not depend on the step size rule at all. Even so, a minimum tuning of LR parameters and some tuning for CP and ESGI are still needed.

B. Example 2

In this case, a job shop scheduling problem from [1, Table IV] is used. The problem consists in 33 machine types and 1 machine for each type, 127 jobs with a total of 184 operations, and a time horizon of 350 units. Each operation has its process time and there are tardiness weights taken fixed values 1.0 or 0.5. There are not earliness weights. Due dates for each job are distributed along the time without overcoming the time horizon

TABLE XI
RESULTS OF EXAMPLE 2 USING LR METHOD WITH DIFFERENT STEP SIZE RULES

Step Size	Num. of iter.	CPU Time (sec)	Primal Cost	GAP (%)
a	108	21	123856	16.21
b.05	500	97	123856	10.53
b.20	500	97	123856	9.15
b.50	428	84	119065	4.98
c	107	20	123856	15.02
d	107	20	123856	15.02
e	107	20	123856	15.02
f	500	97	123856	9.52
g	483	96	118870	4.95
h	464	94	118374	4.42
i	434	84	118752	4.87

TABLE XII
RESULTS APPLYING LR AND CP METHOD (OPTION 1) USING STEP SIZE RULE (h) ON EXAMPLE 2

Iter. CP	Num. of iter.	CPU Time (sec)	Primal Cost	GAP (%)
0	464	94	118374	4.42
40	41	21	116764	4.19
30	55	22	118400	4.97
20	23	16	116961	4.89
10	22	26	117154	4.93
5	16	25	115682	4.93
1	17	40	115756	4.82

but with some negative values, which means already delayed jobs.

The parameters fixed for the stop criteria [(15)–(17)] are

$$\xi = 0.001, \quad n = 500, \quad Gap\ Limit = 5\%.$$

The LR method is applied as explained in Example 1 (Section VII-A) and Table XI shows the results using the same structure than Table I. As in Example 1, tests are named using the step size rule applied. Test a, c, d, and e stop criteria is the proximity of the multipliers of consecutive iterations (15); test $b.05$, $b.20$, and f have reached the total number of permitted iterations (16); and all the others tests have get a GAP less than the fixed GAP Limit (17).

Selected tests are h (the test with the minimum GAP) and $b.20$ (one of the tests which has reached the maximum number of iterations and has a GAP greater than the GAP Limit). For both step size rules, the problem is solved again as in Example 1.

1) Step Size Rule (h): Tables XII–XIV present the results applying Option 1...3, respectively, using step size rule (h). In this case, the use of CP (Option 1) improves the primal cost, and thus, reduces significantly the number of iterations to reach the GAP Limit, as in previous results. Depending on the value of σ, the use of ESGI (Option 2) also improve the results of Option 1. Two different values for σ (ones with the best GAP) have been selected from Table XIII. Option 3 with initial configuration (CP every 40 iterations and $\sigma = 0, 1$) gets worst primal cost and worst GAP than Option 1. Therefore, $\sigma = 0.4$ is also tested using CP every 40 iterations, and it obtains similar primal cost and better GAP than Option 1 with CP every 40 iterations

TABLE XIII
RESULTS APPLYING LR AND ESGI (OPTION 2) USING STEP SIZE RULE (h) ON EXAMPLE 2

σ	Num. of iter.	CPU Time (sec)	Primal Cost	GAP (%)
0.1	305	67	118191	4.33
0.2	461	97	118441	4.46
0.3	500	104	123856	9.24
0.4	359	81	118245	4.33
0.5	500	107	123856	9.23
0.6	399	87	118415	4.46
0.7	500	110	123856	9.22
0.8	500	108	123856	9.21
0.9	500	110	123856	9.22

TABLE XIV
RESULTS APPLYING LR, CP, AND ESGI (OPTION 3) USING STEP SIZE RULE (h) ON EXAMPLE 2

Iter. CP	σ	Num. of iter.	CPU Time (sec)	Primal Cost	GAP (%)
40	0.1	41	20	117213	4.56
40	0.4	41	21	116800	3.99

TABLE XV
RESULTS APPLYING LR AND CP METHOD (OPTION 1) USING STEP SIZE RULE $(b.20)$ ON EXAMPLE 2

Iter. CP	Num. of iter.	CPU Time (sec)	Primal Cost	GAP (%)
0	500	97	123856	9.15
40	41	19	116069	4.99
30	64	24	116742	4.95
20	56	23	115963	4.98
10	27	20	115892	4.94
5	26	21	115521	4.66
1	17	36	115430	4.99

TABLE XVI
RESULTS APPLYING LR AND ESGI (OPTION 2) USING STEP SIZE RULE $(b.20)$ ON EXAMPLE 2

σ	Num. of iter.	CPU Time (sec)	Primal Cost	GAP (%)
0.1	500	108	123856	9.15
0.2	184	39	118483	4.58
0.3	500	106	123856	9.15
0.4	406	85	118274	4.25
0.5	500	105	123856	9.15
0.6	368	77	118166	4.16
0.7	500	107	123856	9.14
0.8	500	108	123856	9.15
0.9	500	110	123856	9.14

(Table XII). The meaning of these results is that the dual convergence using the step size rule (h) is pretty good, and primal convergence can be significantly improved applying CP.

2) Step Size Rule $(b.20)$: Tables XV–XVII present the results applying Option 1...3, respectively, using step size rule $(b.20)$. The initial solution just using LR and the step size rule $(b.20)$ is not good (maximum number of iterations reached and GAP greater then GAP Limit). Even so, Option 1 results in Table XV are quite similar than results in Table XII in terms of number of iterations, CPU time and GAP; and besides, primal costs of Table XV are better than primal costs of Table XII. Option 2 gets

TABLE XVII
RESULTS APPLYING LR, CP, AND ESGI (OPTION 3) USING STEP SIZE RULE ($b.20$) ON EXAMPLE 2

Iter. CP	σ	Num. of iter.	CPU Time (sec)	Primal Cost	GAP (%)
30	0.2	64	24	116742	4.95
30	0.6	64	24	116742	4.95

similar results than previous examples, and Option 3 improves the LR performance and the performance of Option 2. However, Option 3 does not improve the performance of Option 1, and Table XVII presents the results for two tests using CP every 30 iterations and two values of σ, and they are equal than the result using CP every 30 iterations in Table XV. In this case, arbitrary number of iterations and σ values have been chosen for Option 3. Some other configurations have been also tested but not included in Table XVII (Option 3) because the obtained results are the same than in Option 1.

C. Example 3

Ten different instances were generated to test the proposed approaches using larger problems (in number of operations) than the ones in previous examples, but with similar characteristics. The problem consists in ten machine types with a randomly generated number of machines from 1 to 3 in each one. One hundred fifty jobs with a randomly generated number (from 1 to 10) of operations, and an average total of 825 operations. Each operation with a number of alternative machine types randomly generated between 1 and 3, and these alternative machine types are randomly generated between 1 and 10. The processing time of each operation on an alternative machine type is set to a nominal processing time (generated according to a uniform distribution $U[1, 5]$) multiplied by a rate (generated according to a uniform distribution $U[0.8, 1.2]$). Time horizon is fixed in 750, due dates are distributed along the time (depending on the time process of each job) without overcome the time horizon but with some negative values (already delayed jobs). Tardiness and earliness weights of all jobs are set to 1 and 0.1, respectively.

The parameters fixed for the stop criteria [(15)–(17)] are

$$\xi = 0.001, \quad n = 300, \quad Gap\ Limit = 5\%.$$

Step size rules, number of iterations to apply CP, and σ are randomly fixed, and LR parameters are the same for all the instances, without particular tuning depending on the problem, which decrease the quality of the LR solutions. This tuning is not made because the target is to show the benefits of using CP and ESGI approaches together with LR to obtain better solution. Table XVIII present the results for the ten instances. Each row corresponds to an instance and they will be called B01 to B10, following the row order. Two test were made for each instance: LR test and LR, CP, and ESGI test. Column 1 is the step size used for the instances, column 2 (*Iter. CP*), and column 3 (σ) are just used for the LR, CP, and ESGI test, and they are 0 for the LR test. Columns 4–7 include the results from both test separated by a dash: LR/LR,CP, and ESGI.

Results of Table XVIII show that Primal Cost is improved for all instances but B08 when LR, CP, and ESGI is applied. Regarding the other parameters, results can be classified in 3 types: *A)* when the reached stop criteria is the number of iterations for both tests (case of B02, B03, B05, B06, and B09, with white background in Table XVIII); *B)* when the reached stop criteria is the umber of iterations for the LR test and the gap for the LR, CP, and ESGI test (case of B07 and B10, with light gray background); and *C)* when the reached stop criteria is the gap for both tests (case of B01, B04 and B08, with dark grey background). Multipliers criteria has not been reached for any instance. Even half of the results are of type *A*, authors consider that most of results could be of type *B* or *C* if some tuning depending on the instances would be made.

TABLE XVIII
RESULTS OF EXAMPLE 3

Step Size	Iter. CP	σ	Num. of iter.	CPU Time (sec)	Primal Cost	GAP (%)
i	30	0.6	130/61	640/382	229064/227461	5.00/4.45
c	40	0.8	300/300	1771/2479	231128/224429	9.32/6.37
b.20	40	0.7	300/300	1231/2009	376142/364261	8.80/5.49
e	20	0.5	139/61	591/280	181740/181006	5.00/4.96
b.05	30	0.9	300/300	1222/1739	293922/286826	9.48/6.87
b.50	20	0.9	300/300	1299/1982	297769/290053	9.20/6.36
f	10	0.8	300/171	1532/971	168204/165605	5.80/4.58
d	10	0.2	80/31	423/169	237842/237912	4.14/4.79
g	40	0.4	300/300	1330/1975	188293/182638	8.54/5.65
h	10	0.4	300/241	1437/1281	216349/214580	5.56/4.87

Depending on these defined types, it can be seen that, for results of type *A*, the average gap improvement from using LR to using LR, CP, and ESGI is almost of 3 units. For example, B02 gap is 9, 32% using LR, and 6, 37% using LR, CP, and ESGI. However, LR, CP, and ESGI is too time consuming for these types of results, because all the jobs are optimized every time ESGI try to set a new point with better dual cost; therefore, there are more optimizations during the same amount of iterations, which increases the CPU time. For example, the LR method gets a solution for B02 in 1771 s, and LR, CP, and ESGI in 2479 s, 40% of CPU time increase. Type *B* results already show that CP and ESGI approaches improve the convergence of LR obtaining better gaps and less number of iterations and CPU time. Finally, results of type *C* really show the convergence improvement when both approaches are used together with LR. The gap of both tests (LR and LR, CP, and ESGI) are $\pm 1\%$ for results of type *C*. However, the gap criteria is reached faster when applying CP and ESGI. In fact, LR, CP, and ESGI is from 40% (instance B01) to 60% (instance B08) faster than LR. For example, instance B08 gaps are 4, 14/4, 79, but LR reaches the gap after 80 iterations and 423 s, and LR, CP, and ESGI after 31 iterations and 169 s. Number of iterations is improved by 61% and CPU time by 60%.

The conclusions of Example 3 is that the approaches improve the convergence of the LR method by at least 40% less iterations and CPU time if the number of iterations of the stop criteria is big enough (depending on the problem).

The research idea is to develop a method able to obtain better solution than the LR method applying different approaches together just once. However, if there is the possibility to make several tests as made in the first two examples of this paper, it would be helpful to use the best dual and primal solution of all the tests to generate the final solution and the final GAP. In this case, the best primal solution of Example 1 has cost 634 and

the best GAP is 1.60% due to the best dual cost is 624. For Example 2, the best primal and dual costs are 115 430 and 113 482, respectively, and the GAP is 1.72%. If several tests can not be made (as in Example 3) and the number of iterations is not a strong criteria, LR, CP, and ESGI applied together give better performance than just using LR.

VIII. Conclusion

This paper presents a new approach for the LR method (using SG) to improve its performance speeding up the convergence. The ESGI approach has been implemented and it improves the convergence of the LR method by speeding up the dual cost convergence. At the same time, the LR and CP method presented in [5] has been updated achieving better performance, which means less number of iterations and CPU time to get the expected GAP. CP and ESGI are combined together with the LR method and the results are better than applying just the LR method, and better or equal than the results obtained by applying CP or ESGI separately with LR.

ESGI approach could be improved in a future work to determine the step variation (σ) at each iteration instead of to use a fixed value for all iterations. Empirical analysis of the performance sensitivity with respect to σ is considered as a new research to be made in the future.

Future work must also include some algorithms to fix the needed parameters of the LR method and the CP approach without making tests, just using the knowledge on the scheduling problem to solve. Thus, even an operator in an industry, without deep knowledge on the LR method or CP, could obtain good solutions for its scheduling problem.

Besides the improvement of the presented approaches, future work also includes a new implementation of the entire LR method and the approaches on a Constraint Logic Programming framework, which could add a better guide to find as better solutions as possible. Other approaches could be also introduced if they permit the obtaining of better solutions.

The ESGI approach has been developed when SG is used in the LR method. Other methods, as could be Surrogate Subgradient (SSG) [9], are used when problems are too large and optimizing all subproblems takes too much CPU time. SSG does not optimize all subproblems, each iteration, and it implies that ESGI could not be applied as presented in this paper. How to apply the ESGI approach when using SSG is also in the future work that authors will confront.

References

[1] D. J. Hoitomt, P. B. Luh, and K. R. Pattipati, "A practical approach to job shop scheduling problems," *IEEE Trans. Robot. Autom.*, vol. 9, no. 1, pp. 1–13, 1993.

[2] T. Irohara, "Lagrangian relaxation algorithms for hybrid flow-shop scheduling problems with limited buffers," *Int. J. Biomed. Soft Comput. Human Sci.*, vol. 15, no. 1, pp. 21–28, 2010.

[3] M. Wallace, *Constraint Programming*, ser. The Handbook of Applied Expert Systems. Boca Raton, FL: CRC Press, 1998.

[4] E. Tsang, *Foundations of Constraint Satisfaction*. San Diego, CA: Academic Press Limited, 1993.

[5] R. Buil, P. B. Luh, and B. Xiong, "Synergy of lagrangian relaxation and constraint programming for manufacturing scheduling," in *Proc. 6th World Congr. Intell. Control Autom.*, Jun. 2006, pp. 7410–7414.

[6] J. Wang, P. B. Luh, X. Zhao, and J. Wang, "An optimization-based algorithm for job shop scheduling," *SADHANA, J. Indian Acad. Sci., a Special Issue on Competitive Manufacturing Systems*, vol. 22, no. 2, pp. 241–256, Apr. 1997.

[7] H. Chen, P. B. Luh, and L. Fang, "A time-window based approach for job shop scheduling," in *Proc. IEEE Conf. Robot. Autom.*, May 2001, pp. 842–847.

[8] H. Chen, C. Chu, and J. M. Proth, "An improvement of the lagrangian relaxation approach for job shop scheduling: A dynamic programming method," *IEEE Trans. Robot. Autom.*, vol. 14, no. 5, pp. 786–795, 1998.

[9] X. Zhao, P. B. Luh, and J. Wang, "The surrogate gradient algorithm for lagrangian relaxation method," *J. Opt. Theory Appl.*, vol. 100, no. 3, pp. 699–712, 1999.

[10] S. Boyd, L. Xiao, and A. Mutapcic, "Subgradient methods," Oct. 2003. [Online]. Available: http://www.stanford.edu/class/ee392o/subgradmethod.pdf

Roman Buil received the B.S. degree in mathematics and the M.S. degree in industrial engineering–advanced production techniques from the Universitat Autònoma de Barcelona, Barcelona, Spain, in 2002 and 2004, respectively. Currently, he is working towards the Ph.D. degree in industrial engineering—advanced production techniques at the the Universitat Autònoma de Barcelona.

He is an Assistant Teacher at the Department of Telecommunications and Systems Engineering, Universitat Autònoma de Barcelona. His research interests include modeling and simulation methodologies, optimization techniques, production planning and decision making for production planning and logistics. He has been involved in industrial projects working as consultant of DLM-solutions and DLM-aeronautics.

Miquel Àngel Piera received the B.S. degree in computer science from the Universitat Autònoma de Barcelona, Barcelona, Spain, in 1988, the M.S. degree in control engineering from the University of Manchester Institute of Science and Technology, Manchester, U.K., in 1991, and the Ph.D. degree in computer science from the Universitat Autònoma de Barcelona in 1993.

He is a Professor with the Universitat Autònoma de Barcelona since 1994, Head of a Research Group in Modeling and Simulation of Logistic and Production Systems (LogiSim) since 2008. Founder of a Spin-off company focused on modeling and simulation in the logistic field (DLM-Solutions) since 2006. His research interests focus on logistic systems, causal modeling, and discrete-event system simulation.

Peter B. Luh received the B.S. degree in electrical engineering from National Taiwan University, Taipei, in 1973, the M.S. degree in aeronautics and astronautics engineering from the Massachusetts Institute of Technology (MIT), Cambridge, in 1977, and the Ph.D. degree in applied mathematics from Harvard University, Cambridge, in 1980.

Since then, he has been with the University of Connecticut, and currently is the SNET Professor of Communications and Information Technologies and the Head of the Department of Electrical and Computer Engineering. He is also a member of the Chair Professors Group at the Center for Intelligent and Networked Systems, Department of Automation, Tsinghua University, Beijing, China. He is interested in planning, scheduling, and coordination of design, manufacturing, and supply chain activities; configuration and operation of elevators and HVAC systems for normal and emergency conditions; schedule, auction, portfolio optimization, and load/price forecasting for power systems; and decision-making under uncertain or distributed environments.

Dr. Luh is Vice President of Publication Activities for the IEEE Robotics and Automation Society, an Associate Editor of IIE *Transactions on Design and Manufacturing*, an Associate Editor of *Discrete Event Dynamic Systems*, was the founding Editor-in-Chief of the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING (2003–2007), and the Editor-in-Chief of IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION (1999–2003).