A Multi-Agent Architecture for Dynamic Scheduling of Steel Hot Rolling*

P. I. Cowling[1], D. Ouelhadj[2], S. Petrovic[3]

[1] Department of Computing, University of Bradford, Bradford BD7 1DP,UK
[2, 3] School of Computer Science and IT, University of Nottingham,
Jubilee Campus, Nottingham, NG8 1BB, UK.

Abstract

Steel production is a complex process and finding coherent and effective schedules for the wide variety of production steps, in a dynamic environment, is a challenging task.

In this paper we propose a multi-agent architecture for integrated dynamic scheduling of the hot strip mill and the continuous caster. The scheduling systems of these processes have very different objectives and constraints, and operate in an environment where there is a substantial quantity of real-time information concerning production failures and customer requests. Each process is assigned to an agent which independently, seeks an optimal dynamic schedule at a local level taking into account local objectives, real-time information and information received from other agents. Each agent can react to real-time events in order to fix any problems that occur. We focus here, particularly, on the hot strip mill agent which uses a tabu search heuristic to create good predictive-reactive schedules quickly. The other agents simulate the production of the coil orders and the real-time events, which occur during the scheduling process. When real-time events occur on the hot strip mill, the hot strip mill agent might decide whether to repair the current schedule or reschedule from scratch. To address this problem, a range of schedule repair and complete rescheduling strategies are investigated and their performance is assessed with respect to measures of utility, stability and robustness, using an experimental simulation framework.

Key words: steel production, dynamic scheduling, rescheduling, schedule repair, multi-agents.

1. Introduction

For decades, the steel industry has been a powerful symbol of an increasingly global market economy, providing the most important material for many other industries. In recent years, global competitiveness

[1] Peter.Cowling@scm.brad.ac.uk, Tel: +44 (0)1274 234005. Fax: +44 (0)1274 234054.
[2] dxs@cs.nott.ac.uk, Tel: +44 (0)115 846 6521. Fax: +44 (0)115 951 4254.
[3] sxp@cs.nott.ac.uk, Tel: +44 (0)115 951 4222. Fax: +44 (0)115 951 4249.
*Research supported by EPSRC grant number GR/N04225/01, DASH associates and a committee of vice chancellors and Principals ORS award.

and changing customer requirements have further underlined the importance of effective planning, scheduling and control systems. Competitive pressures are moving manufacturers to respond to emerging trends including high quality, low-cost, just in-time delivery, smaller order size for a more precisely defined product, as well as the reduction of product life cycles. Consequently, steel production needs to handle the dynamic nature of demands.

Scheduling production within a modern steel manufacturing facility is a complex task involving a variety of processes each of which interacts with several others in an integrated fashion to produce a final product. The central processes in the chain of steel production are the continuous casters where large slabs of steel are cast, and the hot strip mill where the slabs are transformed into large coils. The hot strip mill and the continuous caster have very different constraints, and need very different scheduling models. Both are subject to real-time events. We focus our attention on the dynamic scheduling of the hot strip mill and its integration with the continuous caster. For many years, research on hot strip mill scheduling has focused on centralised static scheduling using operational research techniques which aim to find optimal solutions (Balas and Martin, 1991; Assaf et al., 1997; Tang et al., 2000). For more realistic and complex models of the scheduling problems heuristics from artificial intelligence have been used such as tabu search (Cowling, 1995; Stauffer and Liebling, 1997; Lopez et al., 1998), genetic algorithms (Chen et al., 1998), and expert systems (Numao and Morishita, 1988; Numao, 1994). All these techniques, whether heuristic or analytical, encounter great difficulties when they are applied to real-world environment, which is characterised by ever-changing task requirements, occurrence of a variety of unexpected events, complicated and dynamic resource constraints, uncertain processing times, and multiple or even conflicting scheduling objectives. In practice, static scheduling is not able to react dynamically and rapidly in the presence of dynamic information not previously foreseen in the current schedule. Furthermore, the centralised approach is especially susceptible to problems of tractability, because the number of interacting entities that must be managed together is large and leads to combinatorial explosion. Often, a detailed schedule is generated over a long time horizon, and planning and execution are carried out sequentially. On the other hand, the inherent nature of steel production environment is distributed. Steel manufacturing is a multi-stage production and it is an interconnected and interdependent

system where both information and decision-making are logically and geographically distributed. Consequently, traditional centralised methods are expensive and not ideally suited to the needs of real-world scheduling systems.

Dynamic scheduling is an important issue in steel production and other manufacturing and employee scheduling domains. It is a challenging new research area (Dorn, 1995; Tang et al., 2001). Recently, Multi-agent systems have proven a major success in solving dynamic scheduling problems and have given answers to the problem of how to efficiently integrate communities of distributed and interactive systems in a dynamic environment. Multi-agent systems have been known to provide capabilities of integration, robustness and reactivity, flexibility, heterogeneity, and autonomy (Parunak 1996, Shen and Norrie, 1999; Cowling et al., 2000; Shen et al., 2001), and appear to be well suited to such complex steel production application. The fundamental objective of multi-agent based-scheduling systems is to provide robustness to disturbances, adaptability and flexibility to rapid changes, and an efficient use of resources.

This paper presents a multi-agent architecture for robust dynamic scheduling of steel hot rolling. Section 2 presents an overview on dynamic scheduling. Section 3 briefly presents the steel production environment. Section 4 describes the multi-agent architecture proposed for the dynamic scheduling of steel hot rolling. Section 5 introduces the measures of utility, stability, and robustness used to evaluate a schedule which is repaired due to the presence of real-time information. Section 6 describes the rescheduling strategies proposed. Section 7 describes the inter-agent cooperation and communication. Section 8 presents experimental results for our simulation prototype. Finally, conclusions are presented in section 9.

2. Dynamic Scheduling

Approaches to production scheduling and rescheduling in a dynamic environment can be classified into three main categories (Shafai and Bruno, 1999): completely reactive approaches, predictive-reactive approaches and robust scheduling. In completely reactive approaches, no firm schedule is generated in advance and decisions are made locally in real-time. The dynamic scheduling problem is viewed as a queuing system by considering each machine as a server. In the queuing system the scheduling decisions

are made as events occur, thus the system cannot be used to create predictive schedules, and so cannot benefit from advances in optimisation technologies. In this situation, simulation has been found to be a desirable technique. Extensive surveys of this topic can be found in Ramasesh (1990), and Suresh and Chaudhuri (1993). In predictive-reactive scheduling, a predictive schedule is generated in advance of execution using available information in the shop floor. When disruptions occur during execution, the predictive schedule needs to be modified in order to take into account the new events. These scheduling/rescheduling methods implicitly treat a dynamic scheduling problem as a series of static problems, which are resolved on a rolling horizon basis. A number of researchers have examined policies of this type (Yamamolto and Nof, 1985; Ovacik and Uzsoy, 1994). Predictive and reactive scheduling may thus be seen as complementary activities. An important issue in which this complementary relationship between predictive and reactive scheduling is highlighted is that of schedule robustness. In the robust scheduling approach, the predictive schedule is built using available information on the disruptions that are likely to occur during execution of the schedule to minimise deviation between the performance measure values of the realised and predictive schedules. Robustness is a desirable attribute of a predictive schedule as it focuses on minimising the effects of disruptions on the performance measures. Wu et al. (1993) considered two possible measures: the deviation from the original job starting times, and the deviation from the original sequence (stability) for one-machine problem in the presence of machine breakdown. The scheduling objective is to minimize shop efficiency (makespan), and at the same time minimise system impact caused by schedule changes. Dorn (1995) used robust scheduling and fuzzy temporal reasoning to represent and propagate schedule uncertainty for a steel-making plant. Cowling and Johansson (2001) proposed two measures, utility and stability, to decide whether to repair a schedule or reschedule from scratch, and surveyed rescheduling and schedule-repair techniques. Utility is the improvement of the objective function resulting from repair, and stability measures the deviation from the original schedule.

During the last few years, successful results have been achieved in using multi-agents to solve complex dynamic scheduling problems. Multi-agents are distributed and autonomous systems that support reactivity, and are robust against failures locally and globally. Agents can locally react to local changes

faster than a centralised system could in an ever-changing environment, and have the ability to cooperate to define a global feasible schedule. The application of multi-agents leads to dynamic scheduling systems that are emergent rather than planned, and concurrent rather than sequential. Multi-agent systems have been applied for developing a wide range of dynamic scheduling applications. Lin and Solberg (1990) modelled the manufacturing floor shop as a market place. Tasks and resources are represented by agents. Each task agent enters the market carrying certain currency and it bargains with each resource agent on which it can be processed. Similarly, each resource agent competes with other agents to get a more valuable task. AARIA (Autonomous Agents for Rock Island Arsenal) (Parunak et al., 1997) developed for an army manufacturing facility is a multi-agent architecture for manufacturing scheduling. Manufacturing resources are encapsulated as autonomous agents and cooperation between the agents takes place through the manager agent. Ouelhadj et al. (1999, 2000) described a multi-agent architecture for dynamic scheduling in flexible manufacturing systems where resources are represented by agents. The resource agents are responsible for scheduling the resources, and they cooperate using the Contract Net Protocol (CNP) (Smith, 1980). Maturana and Norrie (1996) described a mediator architecture for an intelligent manufacturing system which provides virtual organisation through virtual clustering and distributed decision-making through the CNP. A single high-level mediator agent creates dynamically these clusters of heterogeneous agents on an as-needed basis. The activities of the clusters are co-ordinated by the distributed mediator agent. A similar architecture was used in Metaphor II (Shen et al., 2000). In this architecture, the cooperative negotiation among resource agents is realised by combining the mediation mechanism based on hierarchical mediators and the bidding mechanism based on CNP for generating and dynamically maintaining production schedules. Ramos and Sousa (1999) proposed a holonic architecture for scheduling in manufacturing systems in which tasks and resources are represented by holons and used the CNP for scheduling/rescheduling of tasks. Recently leveled commitment contracts were proposed as an extension of the CNP for increasing the economic efficiency of contracts between self-interested agents in the presence of incomplete information about future events. Sandholm (2000) described a leveled commitment contracting protocol for automated contracting in distributed manufacturing. The extended protocol allows self-interested agents to efficiently accommodate future

events by giving the possibility for each agent to decommit from the contract by simply paying a decommitment penalty to the other contract party. A decommitment penalty is assigned to both agents in a contract to be freed from the contract.

3. The Steel Production Environment

In a typical steel production environment, each customer order requires the production of a number of coils with the required physical properties and dimensions, and each coil corresponds to one slab. A coil is described by its delivery due date and its physical proprieties such as: hardness, width and thickness. Steel production involves a range of processes to produce such coils (Figure 1) (Lee et al., 1996; Lopez et al., 1998; Cowling and Rezig, 2000). Raw materials are first melted together in a blast furnace (BF) to produce pig iron. The molten iron is transported to the steel-making shop. The principal stages of steel-making take place in basic oxygen furnace (BOF), ladle treatment (LT), continuous casters (CC) and hot strip mill (HSM). In the basic oxygen furnace, oxygen is passed through the molten iron to reduce the carbon content. The resulting steel is then processed further in the ladle treatment facility to make steel of a certain chemical grade. The molten steel is transported to one or more continuous casters to form solid slabs with different dimensions. The slabs produced are stored in the slabyard (SY). Generally, each customer order can be made at one of several different chemical grades. Orders which may be made at the same grade and have similar dimensions and due dates are grouped and cast in heats, where each heat consists of a fixed weight of molten steel, enough for between five and twenty slabs (depending upon which steel plant we consider). In addition to compatibility of slabs within a heat, consecutive heats must preferably contain chemically similar grades of steel, to be cast at similar widths. Before the slabs are rolled to coils in the hot strip mill, they need to be reheated in the reheat furnaces (RF). The hot strip mill has two sections: the roughing mill and the finishing mill. The roughing mill consists of one stand that reduces the thickness of a slab. The resulting strip is sent to the finishing mills, where there are several rolling stands to progressively reduce the thickness of the steel strip to a required final thickness and width. The thin strip, thousands of feet in length is then coiled. The dimensions of the steel coiled are controlled by a feedback mechanism, and it is highly desirable that consecutive coils should have the

same or similar dimensions. There is a requirement to change finishing rolls every few hours, due to wear and tear, and rolls must be warmed up following a roll change with easy-to-roll coils. There is also a requirement to roll coils of gradually reducing width following the warm up coils, due to marks left on the rolls by the edge of each coil. The width profile of a schedule between roll changes is broadly coffin shaped (Figure 3). The coil thickness and hardness profiles should be relatively smooth.

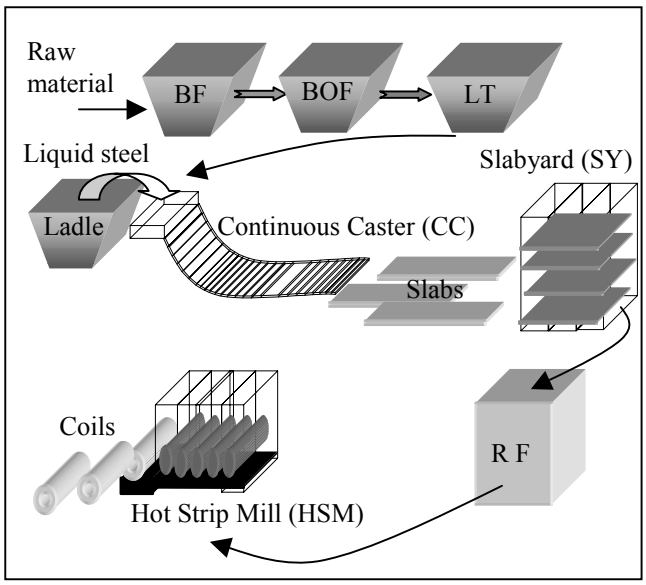

Figure 1. Steel production processes.

4. Multi-Agent Architecture Proposed for Dynamic Scheduling of Steel Hot Rolling

In this paper, we consider only the continuous caster(s), the hot strip mill and the slabyard, whose schedule integration poses one of the greatest challenges, due to the tight and yet flexible linkage between hot strip mill and continuous caster schedules, via the slabyard buffer. Integration and dynamic scheduling of other processes (especially raw material and end product logistics) is also an interesting area of investigation, but the linkages are not so tight, and the level of uncertainty may be so high that a detailed schedule is hard to maintain. The multi-agent architecture proposed (Cowling et al., 2001) is organised as a population of heterogeneous and autonomous agents, each having a set of special skills. Each agent is responsible for the local scheduling of the resource assigned to it, performs optimisation of its own objective function, reacts to real-time events and communicates and cooperates with other agents for global scheduling/rescheduling. Cooperation among the agents for generating and maintaining dynamically production schedules is realised by the exchange of asynchronous messages between the

agents. We use a predictive-reactive methodology to handle real-time events based on the construction of a predictive schedule, which is modified through the dynamic interaction and cooperation of the agents. The architecture involves the following agents (Figure 2): hot strip mill agent (HSM agent), continuous caster agents (CC agents), slabyard agent (SY agent) and User agent.

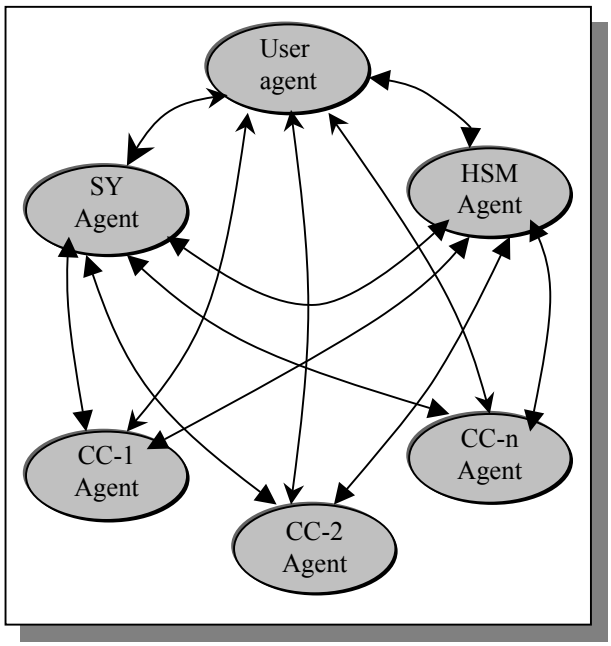

Figure 2. Multi-agent architecture for integrated dynamic scheduling of steel production.

Each agent is composed of:

- **Acquaintance-agents knowledge** includes names of the acquaintance agents and their competencies. This knowledge helps the current agent to select the agents as sub-contractors for processing tasks.
- **Self-knowledge** is the local information used by the agent to execute its own tasks.
- **A model of its local tasks** to be performed.
- **A reasoning module** which plans the actions required to achieve the tasks, e.g. generation of an optimal predictive-reactive schedule.
- **A communication interface** for handling incoming and outgoing messages.
- The HSM agent incorporates a **meta-reasoning module,** which enables it to make a decision on the selection of the scheduling-repair or complete rescheduling strategies in order to react to the presence of real-time events.

4.1. Hot Strip Mill Agent

The HSM agent is responsible for generating optimal predictive schedules, and the dynamic scheduling of the hot strip mill in the presence of real-time events. The hot strip mill scheduling is subject to various constraints (Cowling, 1995; Tang et al., 2000; Cowling et al., 2001). The first scheduling constraint concerns the rollers that suffer wear and tear and need to be replaced at regular intervals. The set of coils produced between two consecutive changes of the finishing mill rollers is called a turn. The set of coils processed between two consecutive changes of the roughing mill rollers is called a shift. Rolling is then organised into shifts, where each shift is a set of turns. The second main constraint is that there should be smooth jumps in coil thickness, hardness and width between consecutive coils in a turn. Smooth jumps in width and thickness, as well as wear on the rollers caused by coil edges require scheduling the coils of each turn in a coffin shape with respect to coil width (Figure 3).

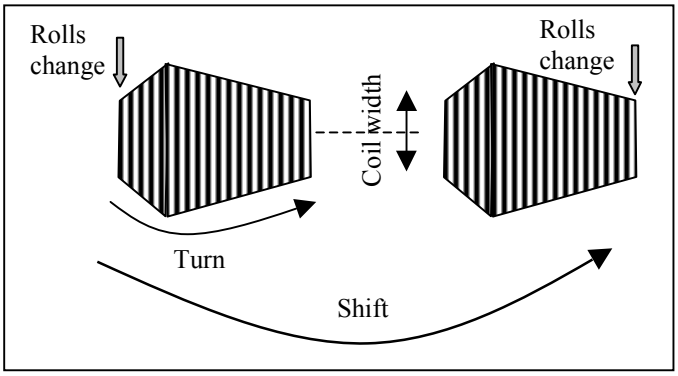

Figure 3. Coffin shape.

The resulting scheduling problem has been formulated as the well-known Prize Collecting Travelling Salesman Problem (PCTSP) (Balas and Martin, 1991; Cowling et al., 2001). The coils are represented by a digraph $G=(N, A)$, where N is the set of nodes and A the set of arcs with weights on both nodes and arcs. Each node $i \in N$ corresponds to a coil, and the weight on the coil_i (score_i) represents priority of the coil to be scheduled. The weight P_{ij} on an arc (i, j) represents the combination of width, gauge and hardness penalties to produce coil_j immediately after coil_i. The penalty is expressed by:

$$P_{ij} = \frac{\alpha(W_i - W_j)^2 + \beta(G_i - G_j)^2}{G_j} \; if \; W_i \geq W_j$$

$$P_{ij} = \frac{\gamma(W_i - W_j)^2 + \beta(G_i - G_j)^2}{G_j} \; otherwise$$

W_i and G_i are the width and gauge of coil$_i$, respectively. $\alpha, \beta,$ and γ are constant parameters, whose values were fixed using knowledge gained from previous investigation with a particular steel mill (Cowling, 1995) as: $\alpha = 1, \beta = 6\times10^4, \gamma = 4$. The scheduling problem is expressed by the function objective:

$$\max \sum_{i=1}^{N} score_i \, Y_i - \sum_{i=1}^{N}\sum_{j=1}^{N} P_{ij} X_{ij}$$

Subject to the following constraints:

X_{ij} and Y_i values are chosen so as to give rise to a path in G.

X_{ij} = 1 if coil j is scheduled right after coil i, X_{ij} = 0 otherwise.

Y_i = 1 if coil i is scheduled, Y_i = 0 otherwise.

To solve this complex optimisation problem, we used a tabu search meta-heuristic (Glover, 1997). The search process starts from Greedy constructive solution. The first coil of the sequence is chosen to be the widest of all the coils. If more than one coil is found to be the widest, then the coil with the highest score is selected. The next coil is chosen to be the one that yields the highest objective value in conjunction with the current coil. The process is repeated until a turn is constructed. Given the Greedy initial sequence, the solution is improved by using three moves: swaps of two coils, insertion of a coil and reversal of a sequence of coils. At each step, all the possible moves involving both scheduled and unscheduled coils are evaluated and the best move is applied until the stopping criterion is reached. The stopping criterion we used is a fixed maximum number of iterations. To avoid cycling on the recently examined solutions, each applied move is subsequently forbidden, or considered tabu, for a certain number of iterations. The tabu moves are kept in a tabu-list, also known as the short-term memory. Each entry in the tabu-list consists of the type of move, the coils involved, the value of the objective function of

the resulting turn, and a count representing the number of iterations for which the move must remain tabu. Each time an entry is added to the tabu-list, its count number is given an initial value, and the count number of each existing tabu move is decreased by one. When the count number reaches zero, the move is removed from the tabu-list, thus becoming free to be considered at the next solution process. In situations where the best possible move is already tabu, an aspiration criterion is applied. Essentially, if the best move yields a better objective function than that of the tabu move, then the best move is applied and as such replaces the tabu move. Conversely, the best move is ignored, and the search continues for the second best move.

4.2. Continuous Caster Agent

The CC agent provides dynamic scheduling and rescheduling of the continuous caster. The main constraints imposed on the schedule of orders are chemical and width compatibility constraints. Slabs to be cast must be grouped in heat lots with each heat cast into several slabs. A casting sequence is a sequence of heats with smooth jumps in width and chemical grade.

4. 3. User Agent

The user agent provides the user interface to the system. It manages and announces the orders to produce, and deals with dynamic changes of order conditions (rush orders, changes in the deadline of orders, etc.).

4.4. Slabyard Agent

This agent is responsible for the management of the slabs produced by the CC agent(s). The SY agent is the mediator agent between the CC agent(s) and the HSM agent. It allows negotiation between the CC agent(s) and the HSM agent to provide a globally good schedule.

5. Robustness, Stability and Utility Measures for Dynamic Scheduling of the Hot Strip Mill

In the present architecture, the HSM agent is concerned with creating predictive-reactive schedules to react to the real-time events which affect the hot strip mill. The real-time events considered are non-

availability of slabs and rush orders. In the presence of unforeseen events, the HSM agent generates schedules that are robust. The HSM agent first constructs a predictive schedule and then modifies the schedule in response to real-time events so as to minimise deviation between the performance measure values of the realised and predictive schedules. In order to make a decision whether to repair the schedule or reschedule from scratch, we defined three measures: robustness, utility and stability. These three measures apply at a high level to the system as a whole, by considering the effects on the output of the process, the hot strip mill schedule. More detailed measures might be developed at the level of each agent (especially for stability) but we will not consider them here. Our results will show that our output measures and agent architecture do produce desirable good results for the system as a whole.

Utility measures the improvement of the original schedule objective due to schedule revision. The utility is the difference between the value of the objective function $F_{dynamic}$ of the new schedule $S_{dynamic}$ after taking into account the real-time information E and the objective function F_{static} of the initial schedule S_{static} before taking into account real-time information. It is expressed by:

$$Utility\ (S_{static}, S_{dynamic}, E, t) = F_{dynamic} - F_{static}$$

Stability measures the deviation from the original schedule caused by schedule revision. The deviation is expressed by the sum of the absolute difference between the original completion time C of the original schedule and the new completion time C' after the occurrence of the real-time event for each coil, which is expressed by:

$$Stability\ (S_{static}, S_{dynamic}, E, t) = \sum_{i=1}^{N} \left| C_i - C_i' \right|$$

The completion time of a coil not in the schedule is expressed as the completion time of the turn plus its processing time. The robustness measure of a schedule S combines the maximisation of the efficiency *utility* and the minimisation of the deviation *stability* from the original schedule. The robustness is expressed as follows, where R is a parameter in the range [0,1]:

$$Robustness(S) = R \times Utility - (1 - R) \times Stability$$

6. Rescheduling Strategies

We defined several strategies for modifying schedules in response to real-time events. On the occurrence of a disruption, the pre-optimised schedule becomes invalid and a new schedule is needed for the set of the remaining coils. Then given the sequence of the remaining coils and a value of R, the HSM agent re-optimises so as to maximise the robustness. For each strategy the utility, stability and robustness measures are evaluated and the strategy which maximises the robustness is applied. The strategies are the following:

a. Rescheduling strategies for non-available slabs

- **Do-nothing (NOT)** ignores the real-time events and deletes the coils corresponding to the non-available slabs.
- **Simple Replacement (SR)** removes the coils corresponding to the non-available slabs from the sequence, and tries to replace them on a one-for-one basis with the best-unscheduled coils so as to maximise robustness (Figure 4).

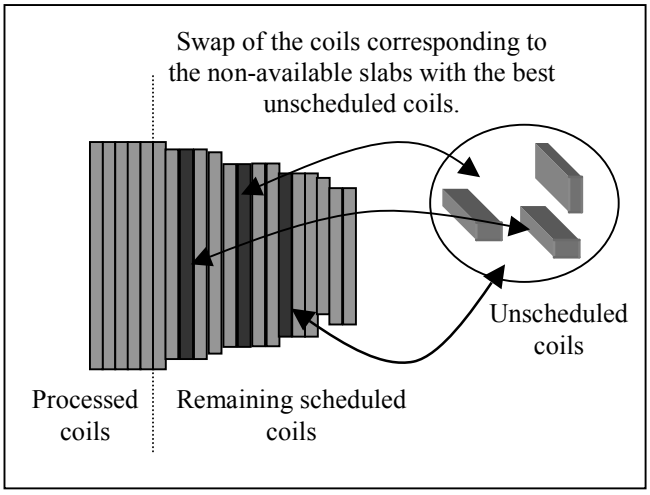

Figure 4. Simple Replacement.

- **Closed Schedule Repair (CSR)** removes the coils corresponding to the non-available slabs and use tabu search to re-optimise the resulting sequence without referring to coils which were not originally scheduled (Figure 5).

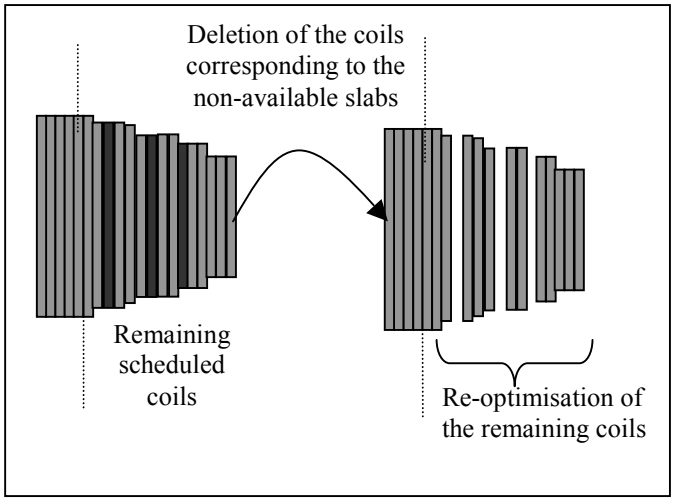

Figure 5. Closed Schedule Repair.

- **Hybrid Closed Schedule Repair (HCSR)** is a combination of SR and CSR strategies. Tabu search starts from the sequence found with SR and tries to re-optimise this sequence without referring to the unscheduled coils.
- **Open Schedule Repair (OSR)** removes the coils corresponding to the non-available slabs and re-optimises the sequence considering the unscheduled coils using tabu search (Figure 6).

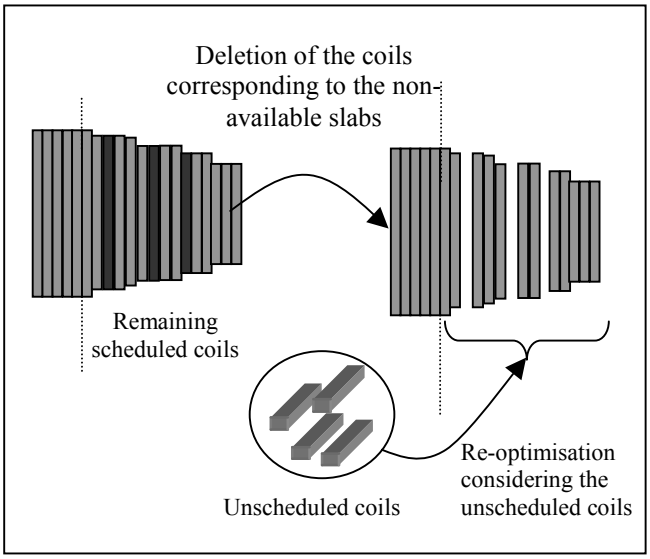

Figure 6. Open Schedule Repair.

- **Hybrid Open Schedule Repair (HOSR)** is a combination of SR and OSR strategies. Tabu search starts from the sequence found with SR and tries to re-optimise this sequence considering the unscheduled coils.

- **Partial Reschedule (PR)** reschedules from the first coil corresponding to the non-available slab using tabu search rather than re-optimising the sequence of the remaining coils.
- **Complete Reschedule (CR)** regenerates a new feasible schedule from scratch.

b. Rescheduling strategies for rush orders

We consider four possible strategies:

- **Do-nothing** evaluates the robustness of the actual schedule ignoring the rush orders.
- **Closed Schedule Repair** replaces the scheduled coils with the rush orders on a one-for-one basis so as to maximise robustness.
- **Open Schedule Repair** inserts the rush orders into the turn and re-optimises the sequence using tabu search considering all available coils.
- **Complete Reschedule** reschedules from scratch taking into account the rush orders which may be added to the new schedule.

7. Inter-Agent Cooperation and Communication

Cooperation among the agents for generating and maintaining dynamically production schedules is realised by the exchange of asynchronous messages between the agents. Communication can be point-to-point (between two agents), broadcast (one to all agents), or multicast (to a selected group of agents). The HSM agent receives a request from the User agent to produce the coils. After receiving the message, it uses a tabu search heuristic to find a good schedule and delegates to the SY agent the task of producing the slabs of the turn. The SY agent sends an announcement message to the CC agent(s) to produce the slabs required for the turn. When certain slabs from the original schedule can no longer be produced from the CC agent due to production failures (raw material fails to arrive on time, slabs manufactured fail to meet right specifications, etc.), alert messages describing the real-time events on the non-available slabs are sent from the SY agent to the HSM agent in order to react to these events as described in section 5.

The communication interface of an agent is composed of several methods for treating all incoming and outgoing messages, and a message queue for storing incoming messages. We have developed a simple language that supports communication between the agents, which contains two categories of messages:

assertions and cooperation messages. Assertion messages include *alert* messages which inform the acquaintance agents about the occurrence of real-time events (non-available slabs, rush orders, resource in failure), and *information* messages to send or request information from acquaintance agents (progress of the production of the coils or slabs). Cooperation messages include *announcements and requests* for tasks which should be performed. Each message is formatted using the eXtensible Mark-up Language (XML). The structure of a message is as follows:

```
<Message>
  <MsgID>    Message identifier        </MsgID>
  <MsgType>  Request, announcement     </MsgType>
  <From>     Name of sender agent      </From>
  <To>       Name of receiver agent    </To>
  <Task>     Task to be performed      </Task>
  <Data>     Data of the message       </Data>
</Message>
```

For example the request message sent from User agent to the HSM agent to produce a sequence of coils.

```
<Message>
  <MsgID>    Ref_146_Orders           </MsgID >
  <MsgType>  Request                  </MsgType>
  <From>     User agent               </From>
  <To>       HSM agent                </To>
  <Request>  Produce _Normal_Coils    </Request>
  <Data>
    <Coil>
      <ID>          ID146_114    </ID>
      <Width>       1.27         </Width>
      <Gauge>       0.00285      </Gauge>
      <Length>      7.018        </Length>
      <Weight>      22.24        </Weight>
      <DueDate>     24 08 2002   </DueDate>
      <Carbon>      77           </Carbon>
      <Aluminium>   64           </Aluminium>
      <Score>       0.51         </Score>
    </Coil>
    <Coil>
      <ID>          ID146_123    </ID>
      <Width>       1.225        </Width>
      <Gauge>       0.003        </Gauge>
      <Length>      6.667        </Length>
      <Weight>      21.2         </Weight>
      <DueDate>     28 08 2002   </DueDate>
      <Carbon>      70           </Carbon>
      <Aluminium>   65           </Aluminium>
      <Score>       0.24         </Score>
    </Coil>
    Etc.
  </Data>
</Message>
```

8. Prototype System and Simulation Results

The current software prototype has been developed as a multi-threaded application in Microsoft Visual C++/MFC. The proposed architecture consists of four agents, implemented as C++ objects: User agent, HSM agent, SY agent, and CC agent (Figure 7). The User agent is responsible for introducing the coil orders and the rush orders. The HSM agent performs the dynamic scheduling/ rescheduling of the coils. The SY agent generates real-time events and passes them to the HSM agent to simulate non-available slabs. The CC agent simulates the production of slabs requested from the SY agent. The cooperation between the agents is done using the exchange of asynchronous messages formatted using XML. In order for the agents to capture the needed acquaintance-agents knowledge, as soon as an agent is started, it broadcasts a message to inform any agent within the architecture of its existence. The message contains personal information about the agent together with its competencies. On receiving the message, an agent sends its own details back to the originator if it happens to be its acquaintance, thus completing an exchange of individual profiles.

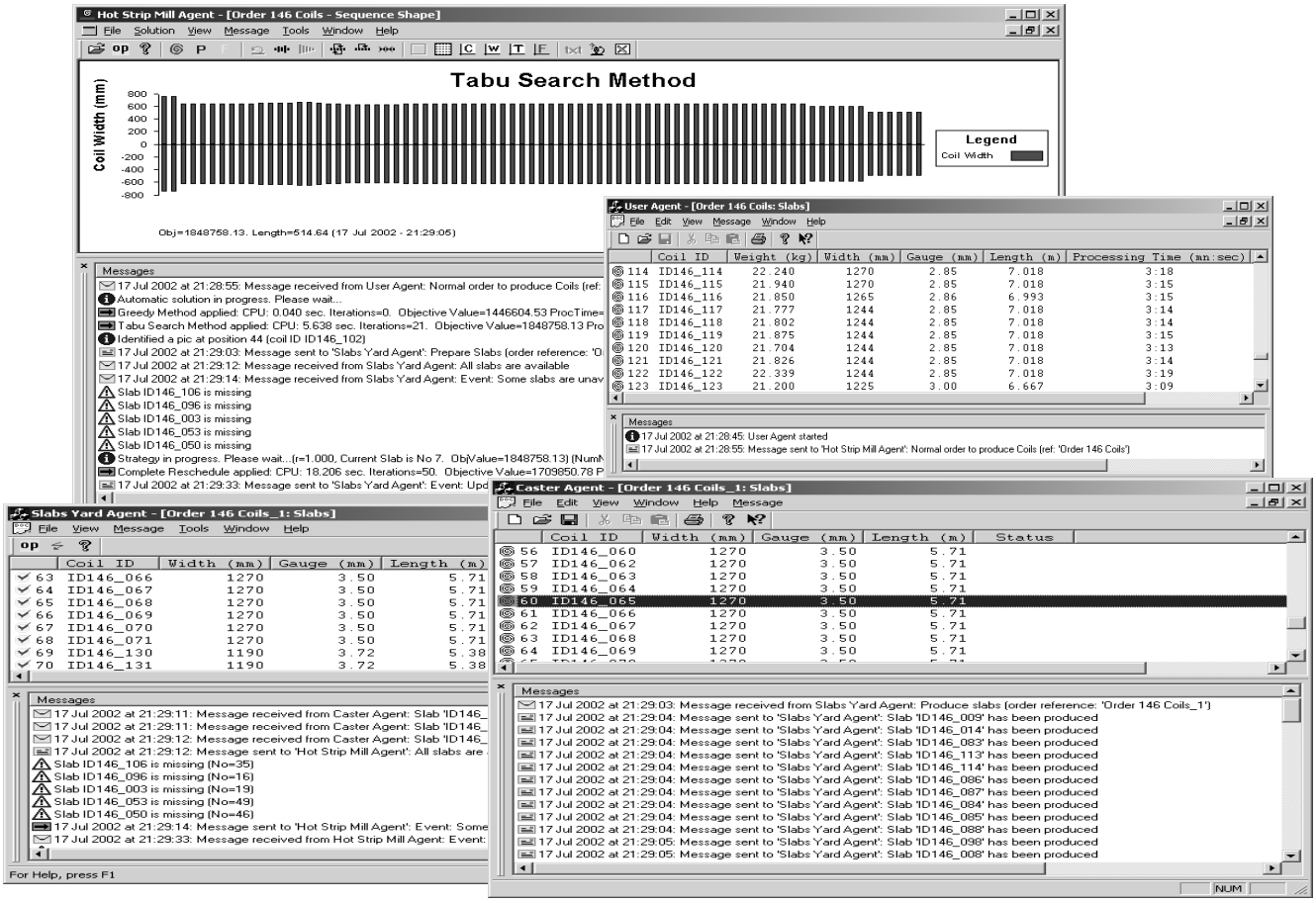

Figure 7. Agents interface.

A set of simulation runs was performed to evaluate the behaviour of the HSM agent in response to real-time events, concerning the non-availability of slabs and rush orders, and investigate the performance of the utility, stability and robustness measures in the presence of real-time events. In order to compare the performance of our strategies, we carried out 5 runs, each consisting of 5 real-time events generated randomly with an initial population of 148 coils provided by a steel manufacturer. Each real-time event specifies the missing slabs and rush orders. The SY agent generates the first event at 10% of the scheduled turn, and the next events every 20% thereafter. For each simulation run, we considered the strategies proposed for different values of R (0, 0.01, 0.25, 0.50, 0.75, 0.95, 1) corresponding to increasing importance of schedule quality, as measured by the objective function, and decreasing importance of schedule stability.

For each event, the HSM agent evaluates the best strategy for different values of R using the robustness function discussed above. Figure 8 summarises our results for the average values of utility, stability, and robustness measures when the strategy which yields the highest robustness is chosen to react to each real-time event. Recall that low values of the stability measure yield low schedule disruption, and high values of the utility measure give a good value of the objective function. These results demonstrate clearly that in an environment where stability should be maintained, the schedule repair strategies give better performance. More precisely, open schedule repair, closed schedule repair, hybrid open schedule repair and hybrid closed schedule repair outperform the other repair strategies in terms of both utility and stability measures. In an environment where we tolerate significant changes in stability in order to gain high utility and a high objective value, complete reschedule, open schedule repair, closed schedule repair and partial reschedule are competitive, but observe that complete reschedule does not dominate the other strategies. In this case, the schedule repair strategies attain similar utility but for better stability.

Figure 9 shows the average frequency of each strategy applied, in response to real-time events, over the five simulation runs. The results demonstrate that irrespective of the importance given to stability or utility, the open schedule repair and closed schedule repair strategies give better performance compared to complete reschedule. However, we remark that the complete reschedule strategy is often selected after a

large number of real-time events have disturbed the schedule. The CPU times for schedule repair strategies were of the order of tenths of a second, whereas those for reschedule were tens of seconds

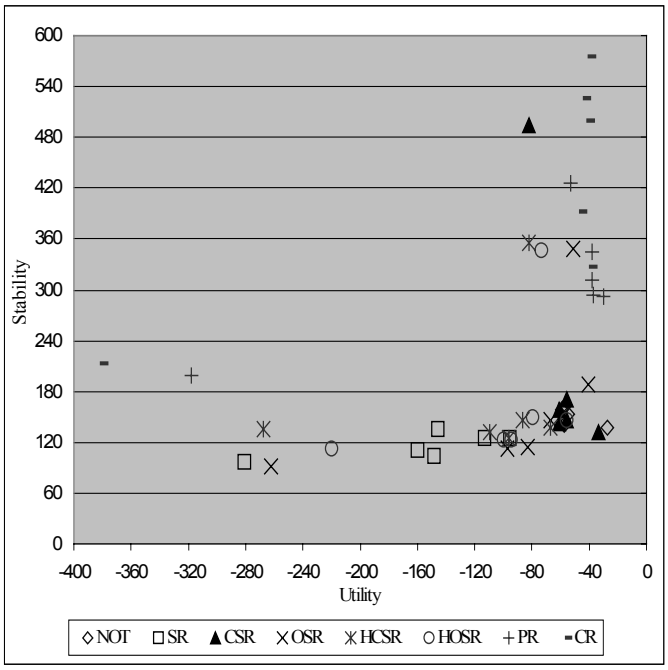

Figure 8. Performance of each strategy when the best strategy is applied in response to real time events for different values of *R*.

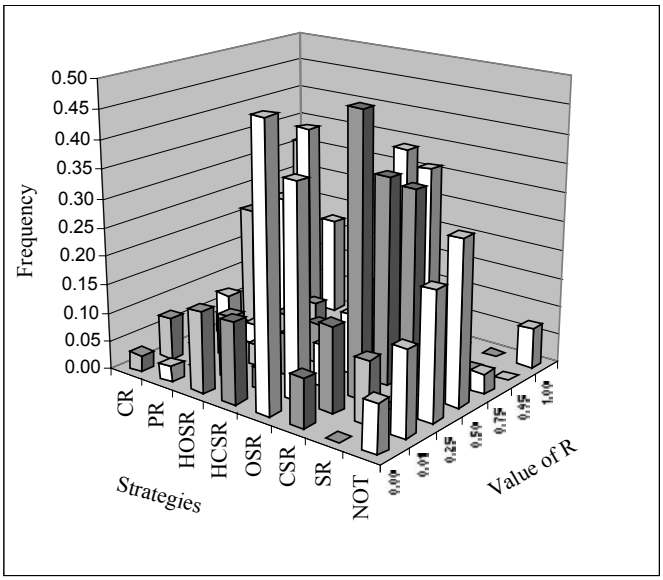

Figure 9. Average frequency of the different strategies.

Figure 10 presents the results of the average frequency of each strategy for three different values of *R* in the presence of a successive sequence of real-time events (event 1, event 2, event 3, event 4 and event 5).

The results demonstrate that the schedule repair strategies open schedule repair, hybrid open schedule repair, closed schedule repair and hybrid closed schedule repair yield the best robustness values, even when stability is neglected (for $R = 1$) when minor changes are applied. However, complete reschedule is often the best strategy to choose when the schedule has been extensively disturbed by a sequence of real-time events, which can be seen in figure 10 where complete reschedule has a high frequency at the level of event 3, event 4 and event 5.

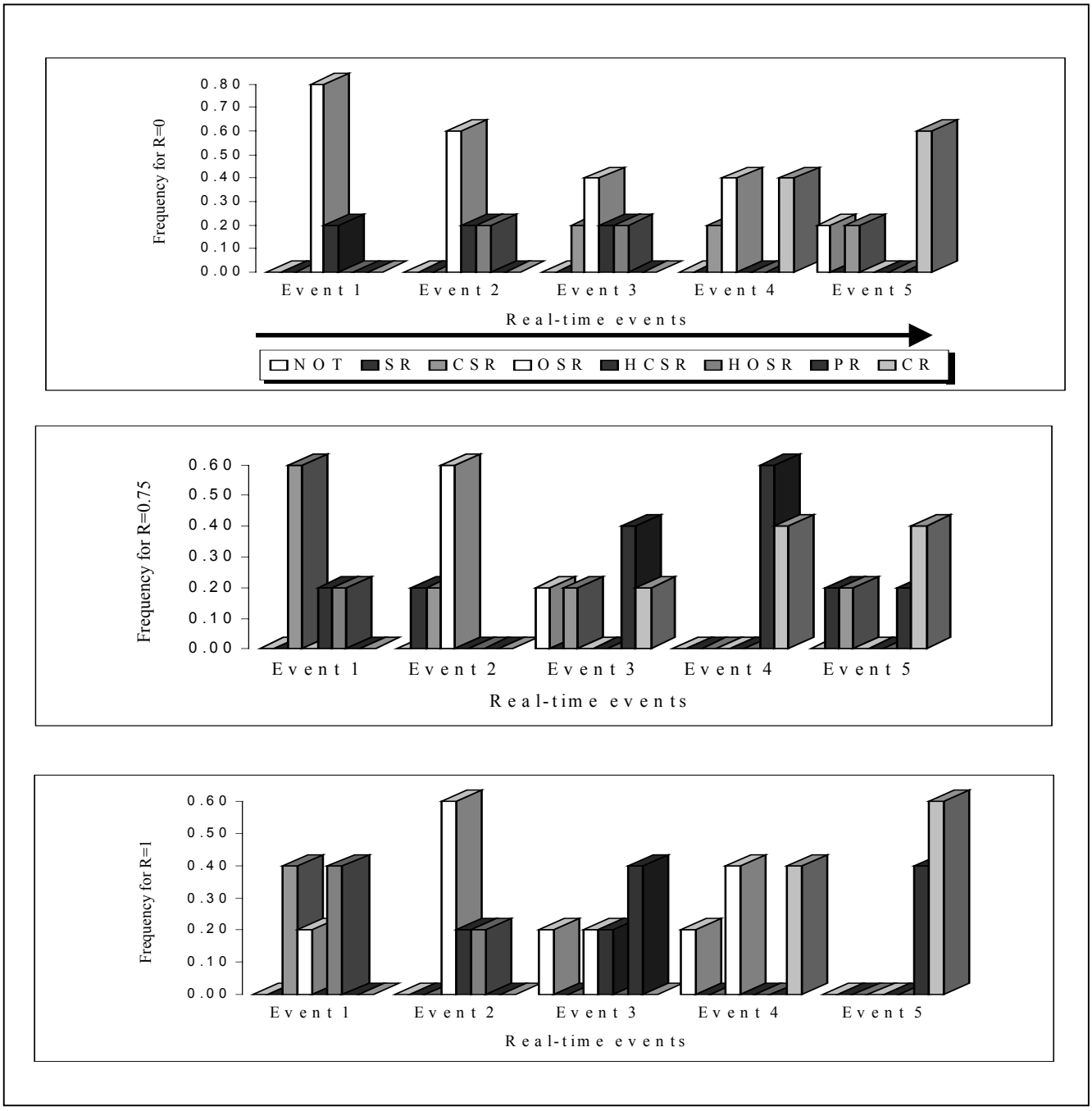

Figure 10. Complete reschedule frequencies in the presence of real-time events for different values of R.

Figure 11 shows the average values of utility, stability and robustness measures when we apply each strategy alone in response to the real-time events. The experimental results show that irrespective of the value of *R*, schedule repair strategies have a consistent behaviour and maintain a better performance in both utility and stability measures compared to complete reschedule.

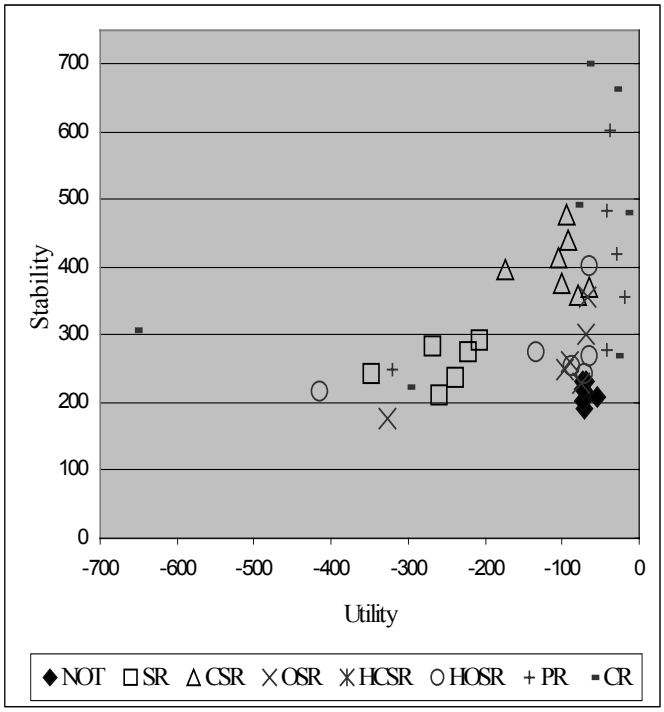

Figure 11. Performance of each strategy when the same strategy is used to react to real-time events.

9. Conclusion

This paper has presented a multi-agent architecture for dynamic scheduling in steel production, particularly for the dynamic scheduling of the hot strip mill. Two key properties of the architecture developed are the integration of scheduling activities of the continuous caster and the hot strip mill, and self-adaptation to real-time events. Our decentralised multi-agent architecture provides a promising approach for efficient enterprise integration and dynamic scheduling within a steel production environment. Robustness to disturbances and adaptability to rapid changes are supported by the local autonomy of the agents and their cooperative behaviour. The autonomous agents use local optimisation meta-heuristics to generate schedules, and respond locally to unpredictable real-time events using the most suitable dynamic rescheduling mechanism. The proposed architecture consists of four dedicated

agents: the User agent, the hot strip mill agent, the slabyard agent and the continuous caster agent. In the simulation prototype, the agents were implemented as C++ objects and their cooperation was carried out using the exchange of asynchronous messages formatted using XML in the communication protocol developed. The scheduling problem for the hot strip mill agent is modelled using the Prize Collecting Travelling Salesman Problem model and solved using a tabu search meta-heuristic. To address the problem of robustness against disturbances, the hot strip mill agent generates predictive-reactive schedules based on three measures (utility, stability, robustness), and a number of schedule repair and complete reschedule strategies which use tabu search meta-heuristics for the search of the best predictive-reactive schedules. The experimental results showed that the schedule repair strategies tend to give better performance in terms of both stability and utility measures. Even in an environment where we tolerate significant changes in stability and require improvements in utility, schedule repair strategies remain competitive. Complete reschedule is, however, often the best strategy after a large number of real-time events have occurred. When we apply the same strategy to react to real-time events, we found that schedule repair strategies have a consistent behaviour and give better performance in both utility and stability measures compared to complete reschedule.

Our results suggest that such a system could provide more stable, higher quality schedules in any environment where there is a significant element of real-time disturbance, as occurs in many manufacturing and personnel scheduling applications.

Future work will continue the investigation of the dynamic scheduling within the continuous caster agent and the improvement of the cooperation mechanism, as well as extending the scale of our work to problems of larger size.

10. References

Assaf, I., Chen, M. and Katzberg, J. (1997) Steel production schedule generation. *International Journal of Production Research*, 35, 2, 467-477.

Balas, E. and Martin, C. H. (1991) Combinatorial optimisation in steel rolling. *Workshop on Combinatorial Optimisation in Science and Technology*, RUTCOR, Rutgers University.

Chen, X., Wan, W. and Xu, X. (1998) Modelling rolling batch planning as vehicle routing problem with time window. *Computers Operations Research*, 25, 12, 1127-1136.

Cowling, P. I. (1995) *Optimisation in Steel Rolling*. In Optimization in Industry, John Wiley & Sons, Chichester, England.

Cowling, P. I. and Johansson, M. (2001) Using real-time information for effective dynamic scheduling. *European Journal of Operational Research*, 139, 2, 230-244.

Cowling, P. I., Ouelhadj, D. and Petrovic, S. (2000) Multi-agent systems for dynamic scheduling. *Proceedings of the Nineteenth Workshop of the UK, PLANSIG 2000*, 14-15th, pp. 45-54.

Cowling, P. I., Ouelhadj, D. and Petrovic, S. (2001) A multi-agent architecture for dynamic scheduling of steel hot rolling. Published in the *Proceeding of the Third International ICSC World Manufacturing Congress*, Rochester, NY, USA.

Cowling, P. I. and Rezig, W. (2000) Integration of continuous caster and hot strip mill planning for steel production. *Journal of Scheduling*, 3, 4, 185-208.

Dorn, J. (1995) Reactive scheduling improving the robustness of schedules and restricting the effects of shop floor disturbances by fuzzy reasoning. *International Journal Human Computer Studies*, 42, 687-704.

Glover, F. (1997) *Tabu Search*. Fred Glover, Manuel Laguna. Boston: Kluwer Academic Publishers.

Lee, H. S., Murthy, S. S., Haider, S. W. and Morse, D. V. (1996) Primary production scheduling at steel making industries. *IBM Journal Research Development*, 40, 2, 231-252.

Lin, G. Y-J. and Solberg, J. J. (1990) Integrated shop floor control using autonomous agents. *IIE Transactions*, 24, 3, 57-71.

Lopez, L., Carter, M. W. and Gendreau, M. (1998) The hot strip Mill production scheduling problem: A tabu search approach. *European Journal of Operational Research*, 106, 2-3, 317-335.

Maturana, F. and Norrie, D. (1996) Multi-agent mediator architecture for distributed manufacturing. *Journal of Intelligent Manufacturing*, 7, 257-270.

Numao, M. (1994) Development of Cooperative Scheduling System for the Steel-Making Process. Eds. *Intelligent Scheduling*, Morgan Kaufman, pp. 607-628.

Numao, M. and Morishita, S. (1988) SCHEPLAN- a scheduling expert for steel-making process. *Proceedings of the International Workshop on Artificial Intelligence for Industrial Applications*, May 25-28, Hitachi, pp. 467-472.

Ouelhadj, D., Hanachi, C. and Bouzouia, B. (1999) A Multi-contract net protocol for dynamic scheduling in flexible manufacturing systems. *ICRA'99, IEEE International Conference on Robotics and Automation*, Detroit, Michigan, USA.

Ouelhadj, D., Hanachi, C. and Bouzouia, B. (2000) Multi-agent architecture for distributed monitoring in flexible manufacturing systems (FMS). *ICRA'2000, IEEE International Conference on Robotics and Automation*, San Francisco, USA.

Ovacik, I. M. and Uzsoy, R. (1994) Exploiting shop floor status information to schedule complex job shops. *Journal of Manufacturing Systems*, 13, 2, 73-84.

Parunak, H. V. (1996) *Applications of Distributed Artificial Intelligence in Industry*. In O'Hare, Jennings (Eds.), Foundation of Distributed Artificial Intelligence, Wiley Inter-science, New York.

Parunak., H. V., Baker, A. D. and Clark, S. J. (1997) The AARIA agent architecture: an example of requirements-driven agent based system design. In *Proceedings of the 1st International Conference on Autonomous Agents*, pp. 482-483.

Ramasesh, R. (1990) Dynamic job shop scheduling: a survey of simulation research. *Omega International Journal of Management Science*, 18, 1, 43-57.

Ramos, C. and Sousa, P. (1999) A distributed architecture and negotiation protocol for scheduling in manufacturing systems. *Computers in Industry*, 38, 2, 103-113.

Sandholm, T. W. (2000) Automated contracting in distributed manufacturing among independent companies. *Journal of Intelligent Manufacturing*, 11, 3, 271-283.

Shafai, R. and Bruno, P. (1999) Workshop on scheduling using practical inaccurate date-Part2: an investigation of the robustness of scheduling rules in a dynamic and stochastic environment. *International Journal of Production Research*, 37, 4105-4117.

Shen, W., Maturana, F. and Norrie, D. H. (2000) Metaphor II: an agent-based architecture for distributed intelligent design and manufacturing. *Journal of Intelligent Manufacturing*, 11, 3, 237-251.

Shen, W. and Norrie, D. H. (1999) Agent based systems for intelligent manufacturing: a state of the art survey. *International Journal of Knowledge and Information Systems*, 1, 2, 129-156.

Shen, W., Norrie, D. H. and Barthes, J. A. (2001) *Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing*. Taylor & Francis, London.

Smith, R. G. (1980) The contract net protocol: high-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, 29, 12, 1104-1113.

Stauffer, L. and Liebling, T. M. (1997) Rolling horizon scheduling in a rolling mill. *Annals of Operations Research*, 69, 323-349.

Suresh, V. and Chaudhuri, D. (1993) Dynamic scheduling: A survey of research. *International Journal of Production Economics*, 32, 53-63.

Tang, L., Liu, J., Rong, A. and Yang, Z. (2000) A multiple travelling salesman problem (MTSP) model for hot rolling scheduling in Baoshan Iron & Steel Complex. *European Journal of Operational Research*, 124, 2, 267-282.

Tang, L., Liu, J., Rong, A. and Yang, Z. (2001) A review of planning and scheduling systems and methods for integrated steel production. *European Journal of Operational Research*, 133, 1, 1-20.

Wu, S. D., Storer, R. H. and Chang, P. (1993) One machine rescheduling heuristics with efficiency and stability as criteria. *Computers Operations Research*, 20, 1, 1-14.

Yamamolto, M. and Nof, S. Y. (1985) Scheduling /rescheduling in the manufacturing operating system environment. *International Journal of Production Research*, 23, 4, 705-722.