Ant Stigmergy on the Grid: Optimizing the Cooling Process in Continuous Steel Casting

Peter Korošec[1], Jurij Šilc[1], Bogdan Filipič[2], and Erkki Laitinen[3]

[1]Jožef Stefan Institute
Computer Systems Department
Jamova 39, SI-1000 Ljubljana, Slovenia
{peter.korosec, jurij.silc}@ijs.si

[2]Jožef Stefan Institute
Department of Intelligent Systems
Jamova 39, SI-1000 Ljubljana, Slovenia
bogdan.filipic@ijs.si

[3]University of Oulu
Department of Mathematical Sciences
P.O. Box 3000, SF-90014 Oulu, Finland
erkki.laitinen@oulu.fi

Abstract

The paper presents a new distributed metaheuristic algorithm in an optimal control problem related to the cooling process in the continuous casting of steel. The optimization task is to tune 18 coolant flows in the caster secondary cooling system to achieve the target surface temperatures along the slab. Sequential search algorithms are proved inefficient for this problem because they take too much time to compute an appropriate solution. For this reason a new distributed search algorithm based on stigmergy perceived in ant colony was developed. The algorithm was run on the Grid that allows us to solve this optimization problem in much shorter time. As a matter of fact, the computation time can be decreased from half a day to a few hours without any decrease in the solution quality.

1. Introduction

Most of the world steel production is nowadays based on continuous casting. This is a complex metallurgical process in which liquid steel is cooled and shaped into semi-manufactures. To achieve proper quality of cast steel, it is essential to control the metal flow and heat transfer during the casting process. They depend on numerous parameters, such as the casting temperature, casting speed and coolant flows. Finding optimal values of process parameters is difficult since different, often conflicting criteria are involved, the number of possible parameter settings is high, and parameter tuning through real-world experimentation is not feasible because of costs and safety risk. Over the last years, however, several computational techniques have been used to enhance the process performance and product characteristics, including knowledge-based heuristic search [3], genetic algorithms [1, 10], particle swarm optimization [18], and evolutionary multiobjective optimization [2].

In this paper we report on numerical experiments in optimizing secondary coolant flows for a casting machine of the Ruukki steel plant in Finland using stigmergic algorithm on the Grid. Here we meet with multi-parameter optimization. Multi-parameter optimization is the process of finding a point in a multidimensional parameter space where a cost function is minimized and constraints satisfied. Most commonly, the cost function contains information about the problem goal and the constraints the solution point has to meet (constrained optimization).

The paper describes the distributed optimization algorithm based on stigmergy and the problem that it solves, provides the results of numerical experiments, and discusses their implications for future work.

2. Ants on the Grid

This section describes the basic concept and major issues pertaining to distributed optimization algorithm

1-4244-0054-6/06/$20.00

based on stigmergy that will be used on the Grid. Stigmergy is a method of communication in decentralized systems where the individual parts of the system communicate with one another by modifying their local environment. For example, ants communicate by laying down pheromone along their trails, so an ant colony is a stigmergic system. The term stigmergy (from the Greek *stigma* = sting, and *ergon* = work) was originally defined in 1959 by the French biologist Grassé [12].

Because of the nature of the ant-based algorithms we first have to discretize a continuous multi-parameter problem and translate it into a graph representation (search graph). Then we use a selected optimization technique to find the cheapest path in the constructed search graph; this path consists of the values of the optimized parameters. For this purpose, we use an optimization algorithm, the routes of which can be found in the ant colony optimization (ACO) metaheuristic [5, 6, 7].

We considered the multilevel approach and its potential to enhance the optimization procedure. The multilevel approach in its most basic form involves recursive coarsening to create a hierarchy of approximations to the original problem. An initial solution is found (sometimes for the original problem, sometimes at the coarsest level) and then iteratively refined at each level. As a general solution strategy the multilevel procedure has been in use for many years and has been applied to various problem areas [22]. We merge stigmergy and the multilevel approach into one method, called the *Multilevel Ant Stigmergy Algorithm* (MASA) [14].

Like many other metaheuristic approaches, the MASA admits direct parallelization schemes and parallelism can be exploited on one or more scales [20]. We applied it on the largest scale where entire search procedures can be performed concurrently. Such implementation, called *Distributed Multilevel Ant Stigmergy Algorithm* (DMASA), is based on parallel interacting ant colonies [16].

2.1 Search graph construction

Search graph construction consists of translation of the discrete parameter values of the problem into a search graph $\mathcal{G} = (V, E)$ with a set of vertices V and set E of edges between the vertices. For each parameter $p_d, 1 \leq d \leq D$, parameter value $v_{\langle d,i \rangle}, 1 \leq i \leq n_d$, $n_d = |p_d|$, represents a vertex in a search graph, and each vertex is connected to all the vertices that belong to the next parameter p_{d+1}. The so-called *start* vertex is also added to the graph. This vertex is connected to all vertices of the parameter p_1 and used as a starting point for all ants.

This way we transform the multi-parameter optimization problem into a problem of finding the cheapest path. Once this is done, we can deploy the initial pheromone values on all the vertices.

2.2 Multilevel approach

Coarsening of the problem representation is done by merging two or more neighboring vertices into one vertex; this is done in L iterations (we call them levels $\ell = 1, 2, \ldots, L$). Let us consider coarsening from level ℓ to level $\ell+1$ at distance d. Here $V_d^\ell = \{v_{\langle d,1 \rangle}^\ell, \ldots, v_{\langle d,n_d^\ell \rangle}^\ell\}$ is a set of vertices at level ℓ and distance d of the search graph $\mathcal{G}$, where $1 \leq d \leq D$. If n_d^1 is the number of vertices at the initial level of coarsening and distance d, then for every level ℓ the equation $n_d^{\ell+1} = \lceil \frac{n_d^\ell}{s_d^\ell} \rceil$ is true, where s_d^ℓ is the number of vertices at level ℓ which merge into one vertex at level $\ell+1$. We therefore divide V_d^ℓ into $n_d^{\ell+1}$ subsets, where

$$V_d^\ell = \bigcup_{k=1}^{n_d^{\ell+1}} V_{\langle d,k \rangle}^\ell, \tag{1}$$

$$\forall i, j \in \{1, \ldots, n_d^{\ell+1}\} \wedge i \neq j \colon V_{\langle d,i \rangle}^\ell \cap V_{\langle d,j \rangle}^\ell = \emptyset.$$

Each subset is defined as follows:

$$V_{\langle d,1 \rangle}^\ell = \{v_{\langle d,1 \rangle}^\ell, \ldots, v_{\langle d,s_d^\ell \rangle}^\ell\}, \tag{2}$$

$$V_{\langle d,2 \rangle}^\ell = \{v_{\langle d,s_d^\ell+1 \rangle}^\ell, \ldots, v_{\langle d,2s_d^\ell \rangle}^\ell\}, \tag{3}$$

$$\vdots$$

$$V_{\langle d,n_d^{\ell+1} \rangle}^\ell = \{v_{\langle d,(n_d^{\ell+1}-1)s_d^\ell+1 \rangle}^\ell, \ldots, v_{\langle d,n_d^\ell \rangle}^\ell\}. \tag{4}$$

The set

$$V_d^{\ell+1} = \{v_{\langle d,1 \rangle}^{\ell+1}, \ldots, v_{\langle d,n_d^{\ell+1} \rangle}^{\ell+1}\} \tag{5}$$

is a set of vertices at distance d at level $\ell + 1$, where $v_{\langle d,k \rangle}^{\ell+1} \in V_{\langle d,k \rangle}^\ell$ is selected on some predetermined principle. Examples are random pick, the most left/right/centred vertex in the subset, etc. The outline of the coarsening procedure pseudo code is as follows:

```
FOR k = 1 TO n_d^{ℓ+1} DO
    v_{⟨d,k⟩}^{ℓ+1} = SelectOneVertex(V_{⟨d,k⟩}^ℓ)
END FOR
```

The goal of optimization is to find the minimum cost path from *start* vertex to a ending vertex (one of the vertices at distance D) in the search graph. There are a number of ants in a colony that all simultaneously start from the *start* vertex. The probability with which

```
Server:
StartAllClients()
WHILE NOT ending condition DO
  IF ReceivedEvaluatedPaths(client) THEN
    BroadcastPaths(all other clients)
  END IF
END WHILE
solutions = ReceiveSolutions&StopAllClients()
bestSolution = Best(solutions)
LocalSearch(bestSolution)
```

```
Client:
graph = GraphConstruction(parameters)
FOR ℓ = 1 TO L DO
  Coarsening(graph[ℓ])
END FOR
GraphInitialization(initial pheromone amount)
FOR ℓ = L DOWNTO 1 DO
  WHILE NOT current level ending condition DO
    FOR all ants in sub-colony DO
      path = FindPath(probability rule)
      Evaluate(path)
    END FOR
    SendEvaluatedPathsToServer(all ants)
    ReceivePathsFromServer(from all other clients)
    UpdatePheromone(all found and received paths vertices)
    DaemonAction(best path)
    EvaporatePheromone(all vertices)
  END WHILE
  Refinement(graph[ℓ])
END FOR
SendSolution(bestSolution)
```

Figure 1. Distributed Multilevel Ant Stigmergy Algorithm (DMASA)

they choose the next vertex depends on the amount of pheromone in the vertices. Ants use a probabilistic rule to determine which vertex to chose next. Ants repeat this action until they get to an ending vertex.

More specifically, ant k in step t moves from vertex i to vertex j with a probability given by:

$$p_{ij,k}(t) = \begin{cases} \frac{\tau_j^\alpha(t)}{\sum_{l \in N_{i,k}} \tau_l^\alpha(t)} & j \in N_{i,k} \\ 0 & j \notin N_{i,k} \end{cases} \quad (6)$$

where α is a parameter that determines the relative influence of the pheromone trail $\tau_j(t)$ in vertex j, and $N_{i,k}$ is the feasible neighbourhood of vertex i.

The parameter values gathered by each ant on its path are now evaluated. Then each ant returns to the *start* vertex and on its way it deposits pheromone in the vertices according to the evaluation result: the better the result, the more pheromone is deposited, and vice versa. After all the ants return to the start vertex, we perform a so-called daemon action, which in our case consists of depositing additional pheromone on currently best path and also some smaller amount on the left and right neighbor paths. Afterwards the pheromone evaporates in all vertices, i.e., the amount of pheromone is decreased by some predetermined percentage in each vertex. The whole procedure is repeated until some ending condition is met.

Because of the simplicity of the coarsening, the refinement itself is very trivial. Let us consider refinement from level ℓ to level $\ell - 1$ at distance d. The outline of the refinement pseudo code is as follows:

```
FOR k = 1 TO n_d^ℓ DO
  FOR each v_⟨d,i⟩^(ℓ−1) ∈ V_⟨d,k⟩^(ℓ−1) DO
    v_⟨d,i⟩^(ℓ−1) = v_⟨d,k⟩^ℓ
  END FOR
END FOR
```

2.3 Distributed implementation

The non-distributed MASA approach presented in [13] is based on a single ant colony. However, in the distributed approach the colony is split into N sub-colonies, where N represents the number of processors. Each sub-colony searches for a solution according to the DMASA algorithm (see Figure 1).

With the use of SendEvaluatedPathsToServer() function the paths with updated pheromone amounts are sent from a client to the server which then with the use of BroadcastPaths() function broadcasts this information to all other clients (sub-colonies). The amount of updated pheromone is determined by the Evaluate() function. On the other hand, the information (paths with updated pheromone amounts) is gathered from other sub-colonies with the use of ReceivePathsFromServer() function. The function UpdatePheromone() deposits pheromone on found and received paths.

Each sub-colony has its own pheromone matrix. With the use of the function UpdatePheromone() a

```
bestSolution = {p_1, ..., p_i, ..., p_D}
change = True
WHILE change DO
  change = False
  FOR i = 1 TO D DO
    IF Evaluate({p_1, ..., p_i + p_i^step, ..., p_D}) < Evaluate(bestSolution) THEN
      change = True
      WHILE Evaluate({p_1, ..., p_i + p_i^step, ..., p_D}) < Evaluate(bestSolution) DO
        bestSolution = {p_1, ..., p_i + p_i^step, ..., p_D}
        p_i = p_i + p_i^step
      END WHILE
    ELSE IF Evaluate({p_1, ..., p_i - p_i^step, ..., p_D}) < Evaluate(bestSolution) THEN
      change = True
      WHILE Evaluate({p_1, ..., p_i - p_i^step, ..., p_D}) < Evaluate(bestSolution) DO
        bestSolution = {p_1, ..., p_i - p_i^step, ..., p_D}
        p_i = p_i - p_i^step
      END WHILE
    END IF
  END FOR
END WHILE
```

Figure 2. Function LocalSearch()

mutual consistency between sub-colony matrices is ensured.

After the clients are stopped with ReceiveSolutions&StopAllClients() function, they return their current best solution and among them the server chooses the best one. On this best solution the server performs the local search. The LocalSearch() pseudo code is given in Figure 2.

2.4 Grid implementation

The DMASA was designed to run on any kind of Grid that uses the TCP/IP protocol for communication. One can notice that the DMASA consists of a parallel part (runs on clients) and a sequential part (local search – runs on the server). Consequently, the search process can only be speeded up in the parallel part while the sequential part remains constant. But we can go further than that. The so-called software pipeline can be used. The pipeline consists of two stages: the first stage is the parallelized part of the algorithm and the second stage is local search. So while the server is running local search, the clients can already search for new solutions. Therefore, new solutions can be acquired as fast as the local search is performed. Of course, with multiple servers, the time needed to acquire new solutions would be further decreased.

3 Continuous steel casting

3.1 Mathematical model

Figure 3 represents a schematic picture of a continuous casting machine. In the continuous casting process the molten steel is poured into a bottomless mold which is cooled with internal water flow. The cooling in the mold extracts heat from the molten steel and initiates the formation of the solid shell. The shell formation is essential for the support of the slab after mold exit. After the mold the slab enters into the secondary cooling area in which it is cooled by water (mist) sprays. The secondary cooling region is divided into cooling zones where the amount of the cooling water can be controlled separately.

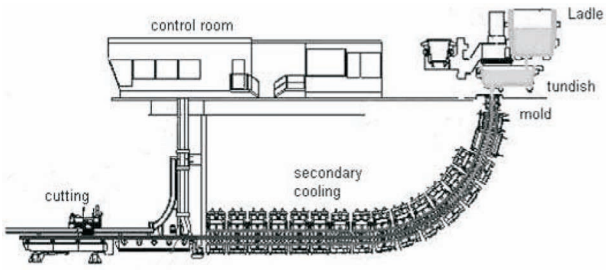

Figure 3. Continuous casting machine

The simulation model calculates the temperature

field of the steel slab as a function of casting parameters. We consider steady state casting conditions, i.e. the parameters are constants in time. We denote the 3D geometry of the slab by $\mathcal{V} = \Omega \times [0, L_Z]$, where $\Omega = [0, L_X] \times [0, L_Y]$ is a 2D cross-section of the slab and L_Z is the length of the strand. Moreover we denote by L_M the length of the mould. We divide the boundary $\Gamma = \partial\mathcal{V}$ into four parts:

$$\begin{aligned}
&\Gamma_0 = \Omega \times \{0\},\\
&\Gamma_N = \{(x,y) \in \partial\Omega : x = 0 \vee y = 0\} \times [L_M, L_Z],\\
&\Gamma_S = \{(x,y) \in \partial\Omega : x \neq 0 \wedge y \neq 0\} \times [0, L_Z) \cup \Omega \times \{L_Z\},\\
&\Gamma_M = \{(x,y) \in \partial\Omega : x = 0 \vee y = 0\} \times [0, L_M].
\end{aligned} \tag{7}$$

The mathematical model for the temperature field $T = T(x, y, z, t)$ of the slab can be written as:

$$\begin{aligned}
&\tfrac{\partial H(T)}{\partial t} + v\tfrac{\partial H(T)}{\partial z} - \Delta K(T) = 0 && \text{in } \mathcal{V} \times (0, t_f],\\
&T = T_0 && \text{on } \Gamma_0 \times (0, t_f],\\
&\tfrac{\partial K(T)}{\partial n} + h(T - T_w) + \sigma\epsilon(T^4 - T_{ext}^4) = 0 && \text{on } \Gamma_N \times (0, t_f],\\
&\tfrac{\partial K(T)}{\partial n} = 0 && \text{on } \Gamma_S \times (0, t_f],\\
&\tfrac{\partial K(T)}{\partial n} = Q && \text{on } \Gamma_M \times (0, t_f],\\
&T(x, y, z, 0) = T^0 && \text{in } \mathcal{V}.
\end{aligned} \tag{8}$$

Here n is the unit vector of outward normal on $\partial\mathcal{V}$, h is the heat transfer coefficient, v is the casting speed, and T_w, T_{ext} are known temperatures. The σ is the Stefan-Boltzmann constant and ϵ is the emissivity. The cooling efficiency Q in the mould is known constant and t_f is the simulation time. $H(T)$ and $K(T)$ are the temperature dependent enthalpy and Kirchoff functions, see [15].

The Eq. 8 is discretized by the finite-element method (FEM) and the corresponding nonlinear set of equations is solved by relaxation iterative methods [8]. More detailed description of discretization and construction of FEM-matrices is presented in [4]. We note that in our method it is sufficient to construct only 2D- and 1D-matrices. Therefore, it is obvious that our model is computationally much more efficient than in the case of using the ordinary 3D-brick elements.

3.2 The coolant flow optimization

The secondary cooling area of the considered casting device is divided into nine zones. In each zone, cooling water is dispersed to the slab at the center and corner positions. Target temperatures are specified for the slab center and corner in every zone. Water flows should be tuned in such a way that the resulting slab surface temperatures match the target temperatures. Formally, a cost function is introduced to measure the differences between the actual and target temperatures. It is defined as

$$\begin{aligned}
c(T) \quad = \quad & \frac{1}{2}\Big(\sum_{i=1}^{N_Z} l_i (T_i^{\text{center}} - T_i^{\text{center}*})^2 + \\
& + \sum_{i=1}^{N_Z} l_i (T_i^{\text{corner}} - T_i^{\text{corner}*})^2\Big)
\end{aligned} \tag{9}$$

where N_z denotes the number of zones, l_i the length of the i-th zone, T_i^{center} and T_i^{corner} the slab center and corner temperatures, while $T_i^{\text{center}*}$ and $T_i^{\text{corner}*}$ the respective target temperatures in zone i. The optimization task is to minimize the cost function over possible cooling patterns (water flow settings). Water flows cannot be set arbitrarily, but according to the technological constraints. For each water flow, minimum and maximum values are prescribed.

Table 1. The target temperatures and water flow intervals

Pos.	Zone no.	T^* [°C]	Parameter	$p_i^{\min}$ [m^3/h]	$p_i^{\max}$ [m^3/h]
	1	1050	p_1	7.1	26.1
	2	1040	p_2	22.8	57.5
c	3	980	p_3	13.3	39.9
e	4	970	p_4	1.5	7.9
n	5	960	p_5	2.7	10.0
t	6	950	p_6	0.8	6.5
e	7	940	p_7	0.7	5.9
r	8	930	p_8	1.0	5.8
	9	920	p_9	1.2	6.2
	1	880	p_{10}	7.1	26.1
	2	870	p_{11}	22.8	57.5
c	3	810	p_{12}	13.3	39.9
o	4	800	p_{13}	1.2	3.5
r	5	790	p_{14}	2.4	4.4
n	6	780	p_{15}	2.4	2.9
e	7	770	p_{16}	0.7	5.9
r	8	760	p_{17}	1.0	5.8
	9	750	p_{18}	1.2	6.2

Table 1 shows an example of the prescribed target temperatures T^* and water flow intervals for continuous casting of a selected steel grade analyzed in this study. The slab cross-section ($L_X \times L_Y$) in this case was 1.70 m × 0.21 m and the casting speed $v = 0.23$ m/s.

4 Experiments and results

4.1 Experimental environment

Evaluation of cooling patterns and their assessment with respect to the cost function (Eq. 9) was done by means of a numerical simulator [15]. Its principal task is to dynamically track the temperature field in the slab as a function of process parameters. It involves a 3D model of the slab and finite element numerical approximation. In this study it was applied under the assumption of steady-state caster operation, and the search for optimal cooling patterns performed in the off-line manner. A single simulator run takes about 25 seconds on a 1.8 GHz AMD Opteron computer.

Table 2. Parameter discretizations

Pos.	Zone no.	Parameter	p_i^{step} [m^3/h]	s_i
	1	p_1	0.1	191
	2	p_2	0.1	348
c	3	p_3	0.1	267
e	4	p_4	0.1	65
n	5	p_5	0.1	74
t	6	p_6	0.1	58
e	7	p_7	0.1	53
r	8	p_8	0.1	49
	9	p_9	0.1	51
	1	p_{10}	0.1	191
	2	p_{11}	0.1	348
c	3	p_{12}	0.1	267
o	4	p_{13}	0.1	24
r	5	p_{14}	0.1	21
n	6	p_{15}	0.1	6
e	7	p_{16}	0.1	53
r	8	p_{17}	0.1	49
	9	p_{18}	0.1	51

Note that the parameter discretization is important part of the optimization process. According to experimental analyze of different discretizations [9], the uniform step size was chosen (see Table 2). The number of possible parameter settings can be obtained as follows. For a parameter p_i from the interval $[p_i^{\min}, p_i^{\max}]$ with step size p_i^{step}, there are $s_i = \lfloor (p_i^{\max} - p_i^{\min})/p_i^{\text{step}} \rfloor + 1$ values possible, and the total number of settings is $s = \prod_{i=1}^{D} s_i$, where D is the number of parameters. This results in $s = 4.7 \cdot 10^{33}$ for step size $p_i^{\text{step}} = 0.1\,\text{m}^3/\text{h}$.

The computer platform used to perform the experiments was an eight-node Grid connected via Giga-bit switch, each node consisted of two AMD Opteron$^{\text{TM}}$ 1.8 GHz processors and 2 GB of RAM. We used a homogeneous Grid, so we could estimate the speed-up. But this is not a requirement for a real-life applications. Clients and server used a standard TCP/IP protocol to communicate between each other.

In the experiments, the stopping criterion for the DMASA was given by the number of solution evaluations. It was set to 768 (eight ants $\times$ eight levels $\times$ 12 climb-downs per level) evaluations per run and this value was chosen considering the computational complexity of the optimization procedure. Other algorithm parameters were set as follows. The DMASA operated with eight levels, at each level ants in sub-colony climbs down the graph until the "level ending condition" is not met (96 evaluations per level in all sub-colonies together). Total number of ants in all sub-colonies is eight, while the number of ants in each sub-colony depends on the number of nodes in the Grid as shown in Table 3.

Table 3. Distribution of ants

	Num. of nodes in the Grid			
	1	2	4	8
Number of sub-colonies	1	2	4	8
Ants per sub-colony	8	4	2	1

Table 4. The optimized coolant flows

Pos.	Zone no.	T [°C]	Parameter	p_i' [m^3/h]
	1	1048.5	p_1	17.2
	2	1039.3	p_2	33.0
c	3	978.3	p_3	20.9
e	4	977.6	p_4	7.9
n	5	960.3	p_5	9.2
t	6	950.0	p_6	5.8
e	7	940.0	p_7	3.1
r	8	930.1	p_8	3.2
	9	919.8	p_9	2.1
	1	880.5	p_{10}	8.4
	2	863.3	p_{11}	22.8
c	3	803.5	p_{12}	15.6
o	4	836.0	p_{13}	3.5
r	5	804.7	p_{14}	4.4
n	6	783.0	p_{15}	2.8
e	7	773.8	p_{16}	1.3
r	8	760.9	p_{17}	1.0
	9	720.6	p_{18}	1.2
The cost of this solution is 9070.4.				

For each distribution (N = 1, 2, 4 or 8), we ran the algorithm 30 times and compared the solution quality and computational time.

4.2 Results

In this section we present and discuss the results of the experimental evaluation of the DMASA algorithm employed with different degrees of distribution (number of nodes used in the Grid).

Table 4 shows the best parameter setting p' found so far [9]. This is the solution obtained in every run regardless of the degree of distribution. For this reason we further concentrate only on the time needed to compute the solution and not on the solution quality.

Table 5. Computation time T [hours:minutes] needed by the DMASA and average speed-up

	Num. of nodes in the Grid			
	1	2	4	8
T_{min}	8:43	5:06	3:57	3:02
T_{max}	10:55	7:56	6:36	5:15
T_{avg}	9:40	6:30	4:54	4:05
Std	0:34	0:33	0:33	0:38
speed-up	1.00	1.49	1.97	2.37

In Table 5 one can see the computational times needed by the DMASA on different number of nodes in the Grid. It is shown that we managed to decrease time quite noticeably from almost ten hours on one node to under four hours on the Grid with eight nodes. The standard deviation remained almost the same through all evaluations, between 33 and 41 minutes.

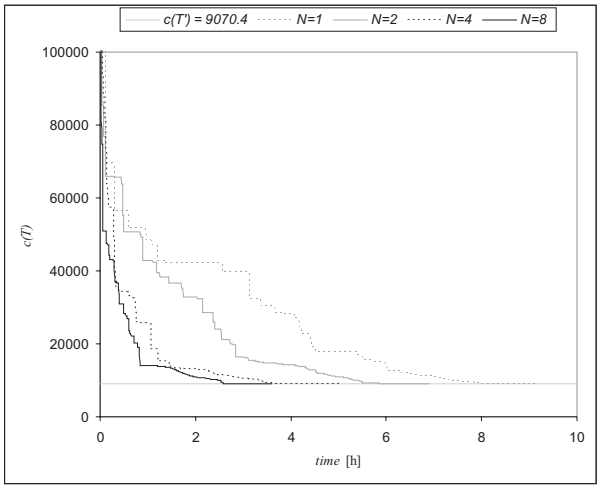

Figure 4. Convergence of the DMASA

Figures 4 and 5 show the convergence and speed-up

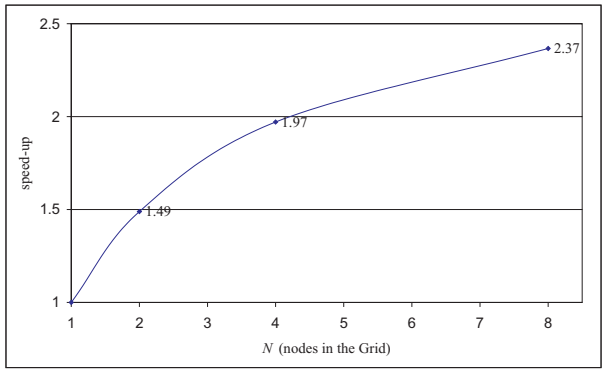

Figure 5. Speed-up of the DMASA

of the DMASA that was achieved with different number of nodes. We can see that the convergence is improved with higher number of nodes and the speed-up is not linear but still quite evident.

5 Conclusions

Optimization of coolant flow settings in continuous casting of steel is a key to higher product quality. It is nowadays to a high degree performed through virtual experimentation involving numerical process simulators and advanced optimization techniques.

In this study of optimizing 18 cooling water flows for a Ruukki casting machine under steady-state conditions, a new distributed search algorithm based on stigmergy perceived in ant colony was applied and its performance evaluated. The results indicate the importance of the applied search space discretization [9] and suggest the construction of a hybrid algorithm to find near-optimal solutions in smaller number of solution evaluations.

We have shown that with the Grid computing the computation time can be drastically decreased (from half a day to a few hours) without any decrease in the solution quality.

It is our interest to further investigate the performance of the proposed DMASA algorithm under different scenarios. The work can be extended with Grid middleware [19], such as the Globus Toolkit [11], Condor [21], or UNICORE (Uniform Interface to Computer Resources) [17].

References

[1] N. Chakraborti, R. Kumar, D. Jain. A study of the continuous casting mold using a Pareto-

converging genetic algorithm. *Applied Mathematical Modelling*, 25(4):287–297, Mar. 2001.

[2] N. Chakraborti, R. S. P. Gupta, T. K. Tiwari. Optimisation of continuous casting process using genetic algorithms: studies of spray and radiation cooling regions. *Ironmaking and Steelmaking*, 30(4):273–278, Aug. 2003.

[3] N. Cheung and A. Garcia. The use of a heuristic search technique for the optimization of quality of steel billets produced by continuous casting. *Engineering Applications of Artificial Intelligence*, 14(2):229-238, Apr. 2001.

[4] R. Dautov, R. Kadyrov, E. Laitinen, A. Lapin, J. Pieskä, V. Toivonen. On 3D dynamic control of secondary cooling in continuous casting process. *Lobachevskii Journal of Mathematics*, 13:3–13, 2003.

[5] M. Dorigo. Optimization, Learning and Natural Algorithms. PhD Thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, 1992.

[6] M. Dorigo, G. Di Caro, L. M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172, Spring 1999.

[7] M. Dorigo and T. Stützle. *Ant Colony Optimization*. Cambridbe, MA, MIT Press, 2004.

[8] C. M. Elliot and J. R. Ockendon. *Weak and Variational Methods for Moving Boundary Problems*. Boston, Pitman Advanced Publishing Program, 1982.

[9] B. Filipič and E. Laitinen. Model-based tuning of process parameters for steady-state stell casting. *Informatica*, 29(4):491–496, Nov. 2005.

[10] B. Filipič and B. Šarler. Evolving parameter settings for continuous casting of steel. *Proc. 6th European Conference on Intelligent Techniques and Soft Computing*, vol. 1, pages 444–449, Sep. 1998.

[11] I. Foster and C.Kesselman. Globus project: a status report. *Future Generation Computer Systems*, 15(5):607–621, Oct. 1999.

[12] P.-P. Grassé. La reconstruction du nid et les coordinations inter-individuelle chez Bellicositermes natalensis et Cubitermes sp. La théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. *Insectes Sociaux*, 6:41–81, 1959.

[13] P. Korošec and J. Šilc. The multilevel ant stigmergy algorithm: An industrial case study. In *Proc. 8th Joint Conference on Information Sciences*, pages 475–478, July 2005.

[14] P. Korošec and J. Šilc. Using stigmergy to solve numerical optimization problems. *International Journal of Computational Intelligence Research*, submitted, 2006.

[15] E. Laitinen. Online control of secondary cooling in steel continuous casting process. In *Proc. COST 526: Automatic Process Optimization in Materials Technology: First Invited Conference*, pages 174–182, May 2005.

[16] M. Randall and A. Lewis. A parallel implementation of ant colony optimization. *Journal of Parallel and Distributed Computing*, 62(9):1421–1432, Sep. 2002.

[17] M. Romberg. The UNICORE grid infrastructure. *Scientific Programming*, 10(2):149–157, 2002.

[18] W. Rongyang, M. Guohui, M. Hongji, C. Ying, X. Zhi. Optimization of the wather rate in secondary cooling zone of continuous casting billet by particle swarm optimization algorithm. In *Proc. Materials Science & Technology 2004*, pages 211, 2004.

[19] B. Schulze, R. Nandkumar, T. Magedanz. Middleware for Grid computing. Special Issue. *Concurrency Computation Practice and Experience*, 16(5):399–400, Apr. 2004.

[20] T. Stützle. Parallelization strategies for ant colony optimization. In *Proc. 5th International Conference Parallel Problem Solving from Nature*, pages 722–741, Sep. 1998.

[21] D. Thain, T. Tannenbaum, M. Livny. Distributed computing in practice: The Condor experience. *Concurrency Computation Practice and Experience*, 17(2-4):323–356, Feb. 2005.

[22] C. Walshaw. Multilevel refinement for combinatorial optimisation problems. *Annals of Operations Research*, 131(1-4):325–372, Oct. 2004.