University of Kent

Kent Academic Repository

Cruz, N. C., Salhi, Said, Redondo, J. L., Álvarez, J. D., Berenguel, M. and Ortigosa, P. M. (2019) *Design of a parallel genetic algorithm for continuous and pattern-free heliostat field optimization.* The Journal of Supercomputing, 75 (3). pp. 1268-1283. ISSN 0920-8542.

Downloaded from

https://kar.kent.ac.uk/67058/ The University of Kent's Academic Repository KAR

The version of record is available from

https://doi.org/10.1007/s11227-018-2404-8

This document version

Author's Accepted Manuscript

DOI for this version

Licence for this version

UNSPECIFIED

Additional information

Versions of research works

Versions of Record

If this version is the version of record, it is the same as the published version available on the publisher's web site. Cite as the published version.

Author Accepted Manuscripts

If this document is identified as the Author Accepted Manuscript it is the version after peer review but before type setting, copy editing or publisher branding. Cite as Surname, Initial. (Year) 'Title of article'. To be published in *Title of Journal* , Volume and issue numbers [peer-reviewed accepted version]. Available at: DOI or URL (Accessed: date).

Enquiries

If you have questions about this document contact ResearchSupport@kent.ac.uk. Please include the URL of the record in KAR. If you believe that your, or a third party's rights have been compromised through this document please see our Take Down policy (available from https://www.kent.ac.uk/guides/kar-the-kent-academic-repository#policies).

Noname manuscript No.
(will be inserted by the editor)

Design of a parallel genetic algorithm for continuous and pattern-free heliostat field optimization

N.C. Cruz [†] · **S. Salhi** [§] · **J.L. Redondo** [†] · **J.D. Álvarez** [†] · **M. Berenguel** [†] · **P.M. Ortigosa** [†]

Received: date / Accepted: date

Abstract The heliostat field of solar power tower plants can suppose up to 50% of investment costs and 40% of energy loss. Unfortunately, obtaining an optimal field requires facing a complex non-convex, continuous, large-scale and constrained optimization problem. Although pattern-based layouts and iterative deployment are popular heuristics to simplify the problem, they limit flexibility and might be suboptimal. This work describes a new genetic algorithm for continuous and pattern-free heliostat field optimization. Considering the potential computational cost of the objective function and the necessity of broad explorations, it has been adapted to run in parallel on shared-memory environments. It relies on elitism, uniform crossover, static penalization of infeasibility and tournament selection. Interesting experimental results show an optimization speedup up to 15x with 16 threads. It could approximately reduce a one year runtime, at complete optimization, to a month only. The optimizer has also been made available as a generic C++ library.

Keywords Genetic algorithm · Parallel computing · Heliostat field optimization · Solar power tower

1 Introduction

As stated in [31], more solar energy arrives to the Earth in one hour than all human consumption in a year. In this context, solar power towers (SPT) are facilities of great interest for large-scale electricity generation due to their high efficiency and scalability [7,26]. Concerning this work, an SPT can be defined

[†] CIESOL Research Centre for Solar Energy, UAL-CIEMAT Joint Centre, Agrifood Campus of International Excellence ceiA3, Dpt. of Informatics, University of Almería (Spain)
E-mail: {ncalvocruz,jlredondo,jhervas,beren,ortigosa}@ual.es
[§] Centre for Logistics and Heuristic Optimization (CLHO), University of Kent (UK)
E-mail: s.salhi@kent.ac.uk
Corresponding author: N.C. Cruz

as a set of high-reflectance mirrors, called 'heliostats', and a radiation receiver on the top of a tower. Heliostats track the apparent movement of the Sun to concentrate the incident radiation on the receiver surface. That energy is then progressively transferred to a heat-transfer fluid to heat it. Finally, it can be applied in a turbine cycle to generate electricity. Further information about SPT can be found in [1,2,5].

The heliostat field can suppose up to 50% of initial investment and 40% of energy loss at operation [3,7]. Thus, designing an optimal field is of high importance to increase the competitiveness of this type of facilities [3,7]. Optimization might aim to maximize the power collected on the receiver as done, for instance, in [16] and in this work. Maximizing the optical efficiency [3,18], which can be seen as a ratio between the intercepted power and the theoretical maximum, is another valid target. Land use [18,23], production price [20] and the trade-off between several targets [30] are also common aspects to consider. Unfortunately, as stated in [7], there is a great lack of information on optimal designs, and field optimization is still an open problem. This is mainly because [6]: i) commercial plants have hundreds, even thousands, of heliostats, which results in large-scale problems with numerous non-convex constraints and ii) the objective function is usually computationally expensive, non-smooth, multi-modal and without an exploitable structure. Hence, as commented in [6,16], most field designs rely on following distribution patterns or iterative placement of heliostats to simplify the problem. The analysis of [14] also includes continuous pattern-free optimization and hybrid methods.

Regarding distribution patterns, the 'radial staggered' layout [13] is one of the most popular examples. It is used, for instance, in [7,18,19]. A more flexible pattern which also relies on radially staggered rows of heliostats was developed in [20]. In [18], it is proposed an innovative bio-inspired pattern based on a Fermat spiral. That layout has become really popular: it has been used, for instance, in [3], and successfully compared to other patterns [17,32]. To attenuate their rigidity, patterns are often applied to generate oversize fields before finally selecting the minimum required heliostats [3,18,20]. Regarding iterative placement of heliostats, the work of [23] is one of the most influencing ones. The work of [28] continues in that way. The method developed in [6] is also iterative but, instead of simulating a pattern-free deployment with a dense grid like [23,28], it adjusts the coordinates of every heliostat directly on the ground. Regarding continuous pattern-free optimization, the gradient-based method of [16] is a good example. Despite considering a continuous search space as [6], heliostats are not iteratively added, but their number is an input. Good ways to optimize the number of heliostats are either their iterative deployment or preliminary oversizing patterns. Finally, according to [14], 'hybrid' methods are those that concatenate several different design strategies. The method of [4], which initially generates a field with any pattern-based approach, is in this category. That would fix important aspects such as the number of heliostats. After that, every one of them would be slightly moved around its zone while trying to improve the field. The improvements observed in [4] state the sub-optimality of some known methods [16].

This paper describes a genetic algorithm which has been designed for continuous and pattern-free field optimization. Aspects such as the receiver size, the tower height and the number of heliostats, which must be optimized too [6,20], should be determined externally. However, its simple and parallel structure aims to be a valuable component of a complete field design methodology. Although genetic algorithms have already been used for heliostat field optimization, the focus is usually on reduced sets of parameters as in [3,17,19, 20]. Similarly, the use of high-performance computing is not a new proposal of this work. Since most common simulations involving SPT are computationally expensive, both the efficiency of methods and the exploitation of modern architectures are generally considered [9]. In fact, it is a key factor when selecting a field design method in [14]. For instance, ray-tracers such as Tonatiuh [25] and SolTrace [27] can use parallel computing to compute the flux map reflected by a certain field on its receiver. In [33], it is presented a ray-tracer for computing the optical field efficiency on GPU at optimization. In [8], three parallelization strategies were proposed for processes based on a similar function definition. Nevertheless, the proposed algorithm aims to be independent of the particularities of the objective function. This can be of major importance if it is not under direct control, e.g., when using external tools and legacy code. Hence, it is easier to ensure an acceptable workload for execution units, and the overhead due to the transition from parallel to sequential code is reduced.

The rest of this paper is as follows: Sec. 2 formalizes the problem. Sec. 3 describes the proposed optimizer. After that, Sec. 4 shows the experiments carried out. Finally, Sec. 5 states the conclusions and some future work. Additionally, complementary material can be found in App. A.

2 Problem statement

Consider a flat surface delimited by i) a minimum (R_{min}) and a maximum (R_{max}) distance from the tower base, which is the origin of coordinates, and ii) a symmetric angular limit from the North, β. Let H be the number of heliostats to place on it. As usual in real fields, they have the same rectangular size $l \times w$, with l and w referring to the length and width, respectively (see Fig. 1). These values, as introduced, are external inputs. Since the surface is flat, the center heights of all the heliostats are the same, and it is enough to define a 2D Cartesian coordinate system on it (see Fig. 1). Every heliostat i can be identified on the field by its central point, (x_i, y_i). Thus, a field of H heliostats can be defined as a vector $F \in \mathbb{R}^{2H}$ according to Eq. (1).

$$F = (x_1, y_1, \cdots, x_H, y_H) \tag{1}$$

There are two main considerations to take into account. First, heliostats cannot trespass the region defined by R_{min}, R_{max} and β. Secondly, heliostats must be able to move without colliding each other. This aspect implies keeping a distance between every pair of heliostats equal or higher than d, which can be computed as $d = \sqrt{l^2 + w^2}$ (see Fig. 1). In fact, although it has been obviated

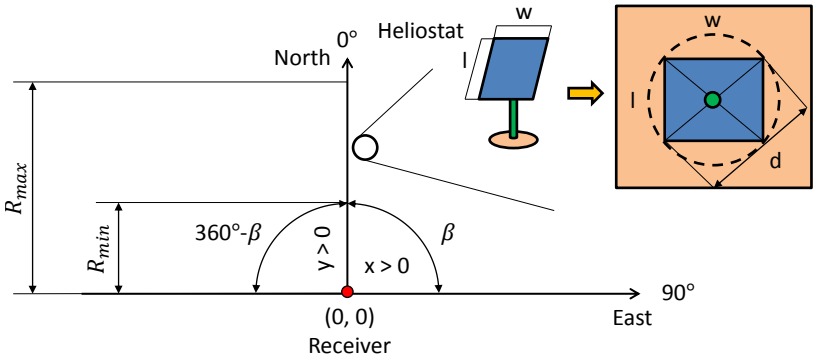

Fig. 1 Representation of the available surface and heliostats

for simplicity, it is frequent to add an extra safety distance to d in order to deal with precision errors and to allow operational purposes [7,20].

Let $P_T(F)$ be the total power effectively reflected by a field F, on the receiver, throughout a fixed set of $T = \{t_1, \cdots, t_T\}$ time instants of interest. Every time instant is defined by an apparent solar position, in angular distance from the ground and the north clockwise (elevation and azimuth angles, respectively), and its incident radiation density (W/m^2). Depending on the application requirements, T can vary from a single one (known as design-point) to many ones encompassing a whole year. In general, the more time instants considered, the more computationally expensive the evaluation of the objective function is, and the more complex the problem becomes.

Considering $P_T(F)$ as the objective function, the target optimization problem can be formalized as expressed in Eq. (2):

$$\begin{cases} \underset{F}{\text{maximize}} & P_T(F) \\ \text{subject to} & R_{min} + \dfrac{d}{2} \leq \sqrt{x_i^2 + y_i^2} \leq R_{max} - \dfrac{d}{2} \; ; \forall i = 1, ..., H \\ & atan(|x_i|, y_i) \leq \left(\beta - asin\left(\dfrac{d}{2\sqrt{x_i^2 + y_i^2}} \right) \right) ; \forall i = 1, ..., H \\ & \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \geq d \; ; \forall i, j = 1, ..., H \; \& \; i \neq j. \end{cases} \tag{2}$$

The problem has $2H$ dimensions, i.e., two coordinates per heliostat. The first and second constraints are linked to keeping the heliostats in the region defined by R_{min}, R_{max} and β. Note that $atan$ returns the azimuth angle of heliostats according to the reference system shown in Fig. 1, which is compared to β modified by the radius to avoid trespassing. By including the absolute value of x, this constraint is expressed in relation to the East side of the field. The third constraint is referred to avoiding collisions among them. There are $H(H-1)/2 + 3H$ constraints to satisfy in total.

In practical terms, defining the configuration of the objective function is also necessary. It will be defined according to Eq. (3) [15,16]:

$$P_T(F) = A \sum_{t=t_1}^{T} I_t \left(\sum_{i=1}^{H} \eta_i(t) \right) \tag{3}$$

A refers to the reflective area of the heliostat model (approximately $l \times w$ (m^2)) and I_t is the incident radiation density at time instant t (kW/m^2). In relation to $\eta_i(t)$, it is the instantaneous efficiency factor of heliostat i at time instant t. This value ranges from 0 to 1, representing the minimum and maximum efficiencies respectively. This factor theoretically depends on: i) the time instant, specifically the apparent position of the Sun, ii) the location of heliostat i in the field and iii) the interaction among heliostats. Its computation implies simulating the behavior of the candidate field. As explained in [3,18, 24], $\eta_i(t)$ is referred to different sources of energy loss at operation. Hence, it is composed by different sub-factors, also in the range $[0, 1]$, which must be ultimately defined. If any of them is not considered, it is equivalent to be assumed 1 (not causing loss). The definition selected for $\eta_i(t)$, shown in Eq. (4), is the same as in [3,15,18].

$$\eta_i = \eta_{cos} \cdot \eta_{sb} \cdot \eta_{itc} \cdot \eta_{aa} \cdot \eta_{ref} \tag{4}$$

Computing $\eta_i(t)$, whose sub-factors are described next, requires a model of the heliostat field. As summarized in [16], several options are available, from developing an ad-hoc model (as in this work and in [3,16,18]) to using an external ray-tracer (not recommended for field optimization due to computational costs [3,18], though). Defining this kind of models is beyond the scope of this work. However, considering its impact on runtime, the implemented model is described below including the references for every one of its components.

First, η_{cos} ('cosine factor') models how the effective reflective area of heliostats is reduced due to their orientation. The reduction factor is the cosine of the angle formed by incident radiation and heliostats. Hence, it is computed as the dot product between the unit vector pointing to the Sun and that normal to the studied heliostat as in [8,18,24]. It must be recomputed for every input time instant.

Second, η_{sb} ('shading & blocking factor') models that heliostats can obstruct either incident (shading) of reflected (blocking) radiation by their peers. It is defined as the ratio between their neither blocked nor shadowed reflective area and the total one [15,18]. To compute it, the strategy proposed in [21] has been applied. It is modeled as a clipping polygon problem: For every heliostat, the four vertices of any other one that could affect it are projected on its plane. Then, all those quadrilaterals are progressively subtracted to the one linked to the reflective surface of the studied heliostat. This idea is depicted in Fig. 2 and has been implemented with the library Clipper [12]. Conflictive heliostats are selected as in [8]: The studied heliostat is wrapped in a circumference as in Fig. 1 and projected towards the Sun and receiver projections over the plane.

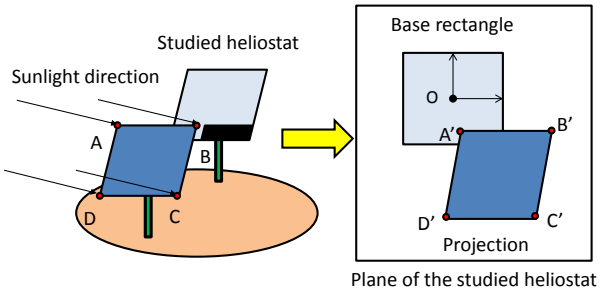

Fig. 2 Shading and blocking estimation based on polygon clipping

If that of any other heliostat can intercept the translation trajectory, it is considered a candidate (of shadowing or blocking when the projection is towards the Sun or receiver, respectively). This factor is the most computationally expensive [3,7,18] and must be recalculated at every time instant too.

Third, η_{itc} ('interception factor') models that the reflected flux maps by heliostats might not fully fall on the desired receiver zone. To compute it, the innovative model proposed in [15] has been used. It estimates the axes of the elliptical shape projected by heliostats, over the receiver plane, depending on its type and relative position. This method, which is based on simple geometry and trigonometry, cannot be as accurate as ray-tracing or convolution methods [3]. However, according to validation shown in [15], it is appropriate for optimization testing and it only needs to be computed once.

Fourth, η_{aa} ('atmospheric attenuation factor') estimates the effect of the atmosphere on the reflected radiation by heliostats. In order to compute it, the analytical model used in [3,8,15,18] has been chosen. It only depends on the distance between every heliostat and the receiver. Hence, this information is also computed and recorded once for every heliostat.

Fifth and last, η_{ref} ('reflectivity factor') represents the fact that heliostats might be unable to grant a lossless reflection phenomenon. It is considered a common fabrication constant as in [18].

3 Method description

Genetic algorithms (GA) [11] are usually applied to solve complex optimization problems. This technique is based on creating a pool of candidate solutions, called population, and mimicking natural evolution on them. Hence, their theoretical principle is not linked to any particular problem but to the abstract evolution of species. This is why they are successfully applied in numerous different problems which, as commented in Sec. 1, includes heliostat field optimization. The definition of GA is generic and its methods must be adapted to the target problem, though. See [22] for further information about GA.

3.1 Overview

In this work, a GA called 'EnGA' has been designed for the problem defined in Sec. 2. Its design not only includes the definition of evolutionary operators but also the explicit possibility of parallel execution in a shared-memory environment. Therefore, the optimizer can benefit from modern high-performance computers. This is of major interest considering that: i) due to the problem complexity, large populations will be required to get competitive results and ii) depending on the selected objective function, its evaluation can be very time demanding, especially for large fields and/or sets of time instants. Additionally, the general parts of the method have been isolated to create a public C++ - OpenMP library (see App. A).

Individuals are vectors in $\mathbb{R}^{2H}$ with an extra field for the fitness of that field design, i.e., the total power reflected on the receiver after simulation (Eq. (3)). In Fig. 3, the structure of individuals is depicted. However, GA are mainly aimed at unconstrained optimization [29] and, as the target problem is constrained, some adaptations must be incorporated. Specifically, the problem will be treated as an unconstrained one but infeasible solutions will be penalized with very low fitness. Penalization will only depend on the degree and number of violations so that the more constraints are not satisfied, the worse fitness is assigned. This strategy, which is quite common to handle constrained optimization problems with GA, is called 'static penalization' [29]. Thus, the constraints in Eq. (2) are ignored at operation, but the objective function is altered to consider them according to Eq. (5):

$$eval(F) = \begin{cases} P_T(F), & \text{if } V = \emptyset \\ -A\left(\sum_{t\in T} I_t\right)\left[\frac{\alpha_i-\alpha_{max_i}}{\alpha_i} v_{i_1} + \frac{(R_{min}+d/2)-m_i}{(R_{min}+d/2)} v_{i_2}\right. & \\ \left. + \frac{m_i-(R_{max}-d/2)}{m_i} v_{i_3} + \frac{d-dist(i,j)}{d} v_{i,j}\right] ; \forall v \in V & \text{otrw.} \end{cases} \quad (5)$$

A is the reflective area of the heliostat model, and I_t is the solar radiation density at time instant t as defined in Sec. 2. α_i is the azimuth angle on the East side of heliostat i, which is computed with function *atan* as included in the first term of the second constraint in Eq. (2). α_{max_i} is the maximum azimuth angle of heliostat i for not trespassing the angular limit, which is computed as the second term of the second constraint in Eq. (2). m_i is the distance between heliostat i and the tower base, i.e., $m_i = \sqrt{x_i^2 + y_i^2}$. V is a set containing, per constraint, either 1 or 0 if it is violated or not, respectively. Specifically: i) v_{i_1} is 1 only if $\alpha_i > \alpha_{max_i}$, ii) v_{i_2} is 1 only if the center of heliostat i is less than $R_{min} + d/2$, iii) v_{i_3} is 1 only if the center of heliostat i is greater than $R_{max} - d/2$, and iv) $v_{i,j}$ is 1 only if the distance between any heliostat i and any other one j is less than d. The boolean structure given to the constraints omits the effect of non-violated ones. Otherwise, their components would counteract the numerical penalization of those affecting ones. Also note that every component is normalized to avoid distortions caused by different numerical ranges. Finally, the factor combining A and every value I_t, which

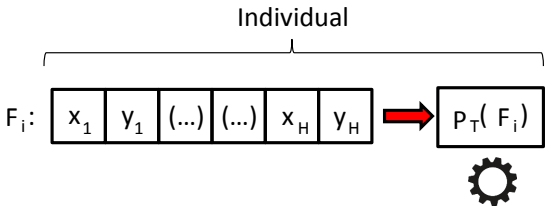

Fig. 3 Structure of individuals

is equal to the maximum obtainable power, scales the order of magnitude of unfeasible solutions to be numerically similar to that of feasible ones.

3.2 Input values

The optimizer, obviating contextual information such as the size of heliostats or the number of threads to deploy (see the details given Sec. 4), gets eight parameters as shown in Alg. 1. *pop* is the population size, which is kept constant. *pairs* is the number of reproduction pairs formed at every cycle while *tourn* is the size of tournaments created to select every progenitor. That information is also used at replacement as tournament selection is applied then too. *mut_ov* and *mut_pb* are the overall probability of launching the mutation procedure on descendants and that of ultimately altering every heliostat, respectively. *init* is the number of heuristic-based, i.e., not random solutions injected to the population at initialization as commented later. *elite* states that the *elite* best solutions in the population will directly survive to the next cycle. Finally, *cycles* is the number of generations.

Algorithm 1: Pseudo-code of the static 'EnGA' genetic optimizer

Input: Int $pop, pairs, tourn, init, elite, cycles$; Real mut_ov, mut_pb
Output: Vector F in $\mathbb{R}^{2H}$

```
1  IndividualSet population, Individual bestInd;          /* Shared among threads */
2  ThreadPool threads = createThreads();                        /* Create a team */
3  threads.runInParallel();                                  /* Thread-local below: */
4  Individual localInd = ∅;                          /* The best I found as a thread */
5  Load range = getChunkSize();                /* Get my zone of work as a thread */
6  population = GenerateInitialPop(range.pop, init)
7  localInd = UpdateBest(range.pop);                  /* Update my view as a thread */
8  < barrier >;            /* The whole population must be ready before looping */
9  for i = 1 to cycles do
10 |   IndividualSet progs = SelectProgenitors(population, range.pairs, tourn);
11 |   IndividualSet desc = Reproduce(progs);
12 |   IndividualSet descMut = Mutate(desc, mut_ov, mut_pb);
13 |   localInd = UpdateBest(desc, descMut);
14 |   popultion < single_barrier >= Replace(population, elite, descMut, tourn);
15 end
16 bestInd < synchr >= UpdateBest&Join(localInd);          /* Best from threads */
17 return bestInd.F
```

3.3 Parallel work-flow

Alg. 1 features a traditional evolutionary loop in which the evaluation of individuals (Eq. (5) in this case) is distributed among threads. In fact, they all stay in the same epoch while a single thread defines the population that survives to the next cycle at line 14 (denoted as '*single_barrier*'). Thus, threads share the cost of evaluating individuals at initialization (line 6), reproduction (line 11) and mutation (line 12) within a consistent population. Similarly, as shown at line 8, creating the initial individuals in parallel requires granting that the whole population could be read by any thread before continuing. Replacement is not computationally expensive, as it does not need new evaluations, and synchronization costs to parallelize it (e.g., to avoid repeated selections) outperforms the hypothetical benefits. Functions $createThreads$, $runInParallel$ and $getChunkSize$ create a thread pool to work on different ranges of the population matrix. This can be considered as a 'static' load-balancing scheme as labeled in Alg. 1 (further comments on this topic are provided in Sec. 3.5). Finally, threads keep a local record of their best solution (lines 7, 13) which is globally updated (tag *synchr*, line 16) at the end. Thus, individuals can be downgraded at mutation (or lost at replacement if $elite = 0$) without risks.

3.4 Behavior of functions

Function $GenerateInitialPop$ (line 6) creates and evaluates *pop* candidate solutions. Every one of them consists of H pairs of coordinates randomly generated in the region defined by R_{min}, R_{max} and β. Thus, they can contain collisions among heliostats, i.e., be partially infeasible. However, as introduced, EnGA can be requested to include *init* heuristic-based individuals. To do so, it uses the algorithm proposed in App. A. It has been specially designed for this purpose, in this work, by adapting the principles of radial staggering [7]. The interested reader could implement any different procedure to create 'promising' individuals dynamically, though. Its recommended features are that i) generating a new individual should not require solving any additional optimization problem, and ii) it should have a certain degree of variability (unless *init* is always set to 1). In fact, if dynamism is not needed, it could be even created a solution pool adapted to any specific requirements. Note that since the cost of creating these individuals is potentially different to random ones, every thread generates an equal fraction of *init*. It is also important to mention that including this kind of solutions influences the whole process. Although the input fields can be completely modified during the search, their initial fitness will be significantly higher than those of random ones. This biases the search and could lead to premature convergence. Then, the fields obtained are likely to be variations of the input ones. This is useful to get acceptable results in a reasonable time. However, non-biased explorations might lead to better results despite the extra effort to converge. Thus, it is a user decision,

and it can only rely on preliminary tests to choose if loading fields, their type and number.

Function $SelectProgenitors$ (line 10) forms the total number of pairs, i.e., $pairs$, from tournament selection. This approach is very popular because it combines uniformity of exploration and adjustable selection pressure. Every progenitor is selected as the best out of a sample formed by $tourn$ participants. This procedure is also applied in function *Replace* (line 14), where a thread selects $pop - elite$ individuals to survive. The $-elite$ term indicates that the $elite$ best individuals are directly moved to the new population (as they could be lost if they did not participate in any tournament). This strategy, called 'elitism', assumes that good solutions can help to improve other ones.

Function *Reproduce* (line 11) gets two descendants from every pair by applying uniform crossover. This method is very popular due to its high rate of mixing that allows a better exploration of the search space. It consists of the following steps: First, a random binary string of length H, called crossover mask, is formed. Every bit has the same probability to be either 0 or 1. Second, a first descendant results from taking the heliostats of its progenitor i at every position where the mask has a 1 and of its progenitor $i + 1$ otherwise. Third, the mask is inverted and a second descendant is obtained by applying the same rules. Any new individual must be ultimately evaluated according to Eq. (5).

Function $Mutate$ (line 12) should make it possible to reach new zones of the search space. To do so, there is a probability mut_ov of attempting the mutation of every descendant. Then, there is a probability mut_pb of randomly relocating every heliostat. Any altered individual must be re-evaluated. Besides, as mutation can downgrade promising solutions, both the altered and original descendants are considered when updating the best result known (line 13). That local record is ultimately used to return the final solution.

3.5 Load-balancing

Finally, the proposed load-balancing is, by default, static. However, Eq. (5) has a different definition for feasible and infeasible solutions. The first type implies simulating T time instants of a field with H heliostats while the second one only depends on the relation between heliostats. Thus, when i) the cardinality of T or H makes a great runtime difference between the type of evaluation and ii) the quantity of feasible and infeasible solutions is very different, this could lead to a serious risk of unbalancing. In that context, dynamic balancing could be implemented by removing line 5 and forcing all threads to get more iterations once they finish their previous ones. Besides, the transition from line 11 to 12 should also be synchronized because threads could try to mutate descendants not created by them, i.e., potentially under creation otherwise. These comments are of interest in any case that the objective function features a conditional nature, though. In fact, the library linked in App. A also offers a dynamic load-balancing mode.

4 Experimentation and results

The experimentation context is based on the field CESA-I, which is part of the PSA solar platform [7]. Thus, the receiver considered is at 86.60 m over the ground. It has a vertical height of 2.45 m and a width of 2.25 m. All the heliostats have a reflective surface with a height of 6.60 m, a width of 6.62 m and a reflectivity factor of 0.8. Their central point is at 3.65 m over the ground. As in the real field, there are 300 heliostats to deploy. The surface available is arbitrarily defined by $R_{min} = 20.0$ m, $R_{max} = 300.0$ and $\beta = 90°$, i.e., all the heliostats should be north of the receiver. It will be considered a single simulation instant (design point): the 21^{st} May at solar noon. According to the models described in [24], then, the solar elevation and azimuth are 72.74° and 180°, respectively. The instantaneous solar radiation density is assumed to be 960 W/m^2. EnGA and the heliostat field model have been implemented in C++ with OpenMP for threading. The execution platform is a computer with 2 Intel Xeon E5 2650 processors (16 cores), and 64 GB of RAM.

EnGA has been first configured to make a broad and not biased exploration of the search space. To do so, the number of initial fields was fixed to 0. Then, every parameter was increased progressively and tested only for 100 cycles for practical reasons. After that, the number of cycles was also increased until the improvements of quality became uncommon. The final configuration was $pop = 1200$, $pairs = 600$, $tourn = 6$, $init = 0$, $mut\_ov = 0.3$, $mut\_pb = 0.05$, $elite = 60$ and $cycle = 10000$. The runtime of EnGA in sequential has been compared to the use of 2, 4, 8 and 16 threads. Fig. 4 plots the speedup achieved with static load balancing and averaged after ten executions. The X axis shows the number of threads and the Y axis contains the speedup. As shown, EnGA achieves a perfect linear acceleration when it runs in parallel. In sequential, EnGA took 176478 s on average while this runtime was reduced to 11156 s with 16 threads. Thus, the same optimization procedure can be executed up to 15.82 times faster. In fact, the average speedup with 2, 4 and 8 threads are 2.03, 4.05 and 8.06, respectively, i.e., marginally higher than the ideal value, which is caused by the stochasticity of EnGA: threads do not necessarily execute the same process.

Regarding dynamic load-balancing, the speedups are very similar. For instance, with 16 threads, it is approximately 15.67. However, the speedup seems to be slightly lower. This is because of the extra synchronization of task dispatching. As commented, the evaluation time of the real objective function is significantly different from that of the penalized version. Numerically, the real function takes 0.015 s while the penalized one requires 0.0005 s, i.e., the real evaluation of candidate fields is 30 times slower than the penalized one. Hence, dynamic load-balancing could be expected to perform better than the static approach. However, after the initial 70-200 cycles of search, the population has multiple feasible solutions that require real evaluations. Thus, potential unbalancing lasts less than 2% of the cycles.

An alternative configuration approach would renounce flexibility and rely on injecting initial fields. To do so, the previous base configuration has been

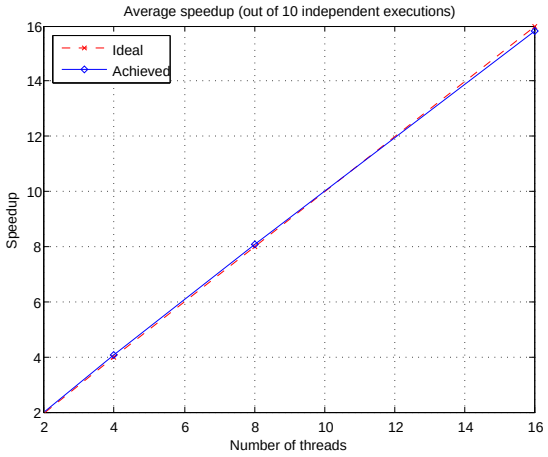

Fig. 4 Speedup achieved with parallelization and static load balancing

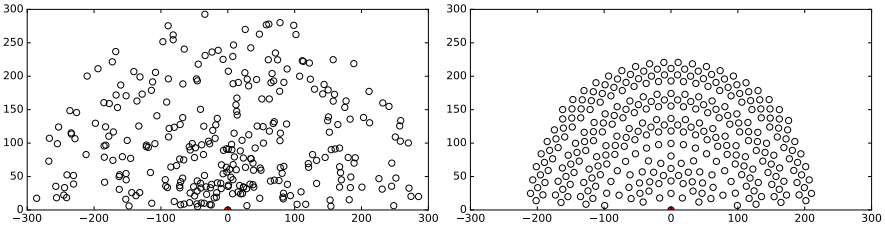

Fig. 5 Examples of initial solutions: random (left) and heuristic-based (right)

altered by setting $init = 60$ (5% of pop). Fig. 5 shows two initial solutions to illustrate the effect of this change. The one on the left has been randomly generated. The other one has been created by EnGA through the knowledge-based strategy also developed in this work (see App. 1). The first one contains multiple collisions among heliostats and requires a significant improvement. In contrast to it, the other one has a robust design that provides EnGA a promising initial solution. This extra help allows reducing the number of cycles to 200. With this base configuration, the sequential version only takes 2676 s on average. The static parallel approach still shows a perfect linear speedup. However, in this case, the dynamic load-balancing version outperforms the other one: its speedup is up to 5% higher when 16 threads are launched. Since the number of cycles is lower, the convergence of the population does not hide the benefit of paying attention to load unbalancing.

After having confirmed the positive effect of parallelization on both configuration approaches, it is interesting to compare how they affect the optimization results achieved by EnGA. Tab. 1 includes the average efficiency of the fields obtained with every configuration, independently of the number of threads and the load balancing style. The first three columns summarizes the main details of each configuration, i.e., they have the same population size but

Table 1 Optimization results of EnGA depending on the configuration approach.

	pop	*init*	*cycles*	**Av. Eff. (%)**	**Av. Seq. Time (s)**
1	1200	0	10000	**69.25**	**176478**
2	1200	60	200	**70.62**	**2676**

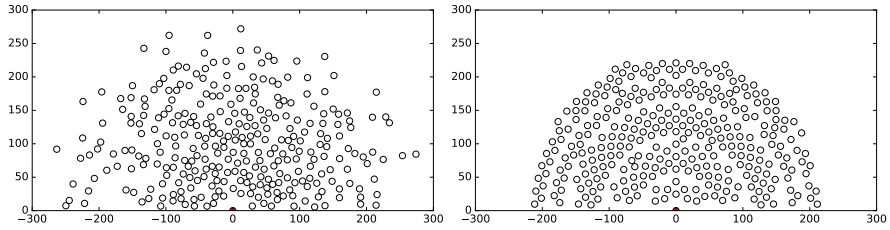

Fig. 6 Fields obtained starting without (left) and with (right) heuristic-based solutions

the second one uses knowledge-base individuals and requires fewer cycles. The last two columns, highlighted in bold, show the average efficiency of the fields obtained in each case and the runtime in sequential, respectively. The first configuration makes EnGA generate fields with an average efficiency of 69.25 % (around 8700 kW on the receiver at design point). This value is near to the maximum value that could be achieved, i.e., 80 %, which is the upper bound implicitly defined by the reflectivity of the heliostats. However, the second one yields fields with an average efficiency of 70.62 %. Thus, those fields are clearly better. Although the numerical difference of efficiency might seem negligible, it can be relevant in this context. Since these values are referred to a single time instant, there is little room for improvement. Besides, the yearly improvements achieved in [18] are very well considered in spite of looking numerically low. Another advantage of the second configuration is that it results in significantly faster executions. Note that parallelization can accelerate both executions up to 16, though. For the sake of completeness, Fig. 6 shows two examples of fields obtained by EnGA when it does not use heuristic-based solutions and when it does. Although the first one tends to accumulate heliostats in the central zone of the field logically, its external zone seems improvable. In contrast to it, the second one has a structure better balanced. As predicted, it is directly based on a heuristic-based solution with modifications.

According to the previous analysis, the use of initial knowledge-based solutions seems advisable for the problem instance. It allows better fields in less time, which would be a critical point when considering multiple time instants. However, as discussed in Sec. 3 and supported by Fig. 6 (right), the fields obtained in this way are mainly one of those initial solutions with some variations. Thus, it is still possible to opt for a high number of cycles to allow more mutations with the hope of achieving better results. Some experiments with this strategy lead EnGA obtaining field efficiencies up to 0.20% higher. Although the runtime becomes approximately equal to that registered with the initial configuration, the best fields have been achieved in this way. Nevertheless, it is always possible to opt for non-biased explorations with more

runtime due to their theoretical interest. Thus, it all depends on the final goals and the availability of time and computational resources.

Back to the computational perspective, although the previous experiments have been defined at design-point, real evaluations for yearly-scoped optimization should consist of, at least, 192 time instants [10,16]. Then, as penalized evaluations can be neglected, the runtime would be amplified by 192. Thus, the sequential version would approximately take 176478 s $\times$ 192 = 33883776 s (more than a year). The parallel version with static load-balancing, which performs better with large numbers of cycles, would only need 11156 s $\times$ 192 = 2141952 s (less than a month). To validate this estimation, four more time instants were added to the previous one and runtime was multiplied by 5 while the speedup was maintained. Moreover, a simpler problem instance of 100 heliostats was studied for 1, 3, 5 and 7 time instants. The runtime was also increased as expected. However, the speedup was slightly reduced (up to 14.04 with 16 threads), but this is because of the lower computational load. Additionally, it is interesting to mention that the efficiency achieved was higher due to the simplicity of the problem (around 75.7% for a single time instant). It was reduced progressively when increasing the number of time instants (around 65.7% for 7 ones). This behavior is consistent with the fact that, as mentioned in Sec. 2, the more time instants, the more difficult that the problem becomes.

5 Conclusions and future work

In this work, the problem of continuous and pattern-free heliostat field optimization has been presented. Although this approach is known to lead to good results, it is not generally applicable due to its conceptual complexity and computational cost. Therefore, it has been designed a parallel genetic algorithm called EnGA to face this situation. Its structure enforces sharing the cost of evaluating candidate solutions among different threads, and independently of the field model used.

Parallelization effectively reduces the overall runtime without affecting resolution capabilities. The speedup can be considered linear featuring a peak of 15.93 with static load-balancing and 16 threads. In fact, due to optimization stochasticity and time averaging, it can be even slightly higher than 16 with knowledge-based solutions, dynamic load-balancing and a low number of cycles. This latter strategy only seems to be useful with that kind of configuration, though. Besides, based on the observed trend, at yearly optimization, more than a year of runtime could be reduced to less than a month by using parallel computing. Hence, wider regions of the search space can be explored for the same runtime. Moreover, the parallelization scope is not linked to any particular objective function but to the optimizer itself. This aspect facilitates granting enough workload for threads and enhances its applicability for different problem definitions. Thus, general comments regarding the underlying context and some aspects to consider, such as potential load unbalancing, have

also been made. In fact, not only a generic library with the optimizer base has been made public, but a new heuristic to generate promising staggered fields has been proposed too.

Future work is related to a new method that is under development to facilitate continuous and pattern-free field optimization. To do so, it states how to divide the problem into simpler instances to be ultimately addressed by any selected optimizer. EnGA will be hence applied to that task.

Acknowledgements This work has been funded by grants from the Spanish Ministry of Economy, Industry and Competitiveness (TIN2015-66680-C2-1-R & ENERPRO DPI 2014-56364-C2-1-R), Junta de Andalucía (P12-TIC301). N.C. Cruz (FPU14/01728) is supported by an FPU Fellowship from the Spanish Ministry of Education. J.L. Redondo (RYC-2013-14174) and J.D. Álvarez (RYC-2013-14107) are fellows of the Spanish 'Ramón y Cajal' contract program, co-financed by the European Social Fund. The authors also wish to thank Juan José Moreno Riado for his technical support.

References

1. Alexopoulos, S., Hoffschmidt, B.: Advances in solar tower technology. WIREs Energy Environment **6**(1), 1–19 (2017)
2. Behar, O., Khellaf, A., Mohammedi, K.: A review of studies on central receiver solar thermal power plants. Renewable and Sustainable Energy Reviews **23**, 12–39 (2013)
3. Besarati, S.M., Goswami, D.Y.: A computationally efficient method for the design of the heliostat field for solar power tower plant. Renewable Energy **69**, 226–232 (2014)
4. Buck, R.: Heliostat field layout improvement by nonrestricted refinement. Journal of Solar Energy Engineering **136**(2), 1–6 (2014)
5. Camacho, E.F., Berenguel, M., Rubio, F.R., Martínez, D.: Control of solar energy systems. Springer (2012)
6. Carrizosa, E., Domínguez-Bravo, C., Fernández-Cara, E., Quero, M.: A heuristic method for simultaneous tower and pattern-free field optimization on solar power systems. Computers & Operations Research **57**, 109–122 (2015)
7. Collado, F.J., Guallar, J.: A review of optimized design layouts for solar power tower plants with campo code. Renewable and Sustainable Energy Reviews **20**, 142–154 (2013)
8. Cruz, N.C., Redondo, J.L., Berenguel, M., Álvarez, J.D., Becerra-Terón, A., Ortigosa, P.M.: High performance computing for the heliostat field layout evaluation. The Journal of Supercomputing **73**(1), 259–276 (2017)
9. Cruz, N.C., Redondo, J.L., Berenguel, M., Álvarez, J.D., Ortigosa, P.M.: Review of software for optical analyzing and optimizing heliostat fields. Renewable and Sustainable Energy Reviews **72**, 1001–1018 (2017)
10. Duffie, J.A., Beckman, W.A.: Solar engineering of thermal processes. John Wiley & Sons (2013)
11. Holland, J.H.: Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. Michigan Press (1975)
12. Johnson, A.: Clipper– an open source freeware polygon clipping library (2012). Available from `http://www.angusj.com/delphi/clipper.php` (Last access in July, 2017)
13. Lipps, F., Vant-Hull, L.: A cellwise method for the optimization of large central receiver systems. Solar Energy **20**(6), 505–516 (1978)
14. Lutchman, S., Gauché, P., Groenwold, A.: On selecting a method for heliostat field layout optimization. In: 2nd Southern African Solar Energy Conference (SASEC) (2014)
15. Lutchman, S.L.: Heliostat field layout optimization for a central receiver. Master's thesis, Stellenbosch University (2014)
16. Lutchman, S.L., Groenwold, A.A., Gauché, P., Bode, S.: On using a gradient-based method for heliostat field layout optimization. Energy Procedia **49**, 1429–1438 (2014)

17. Mutuberria, A., Pascual, J., Guisado, M.V., Mallor, F.: Comparison of heliostat field layout design methodologies and impact on power plant efficiency. Energy Procedia **69**, 1360–1370 (2015)
18. Noone, C.J., Torrilhon, M., Mitsos, A.: Heliostat field optimization: A new computationally efficient model and biomimetic layout. Solar Energy **86**(2), 792–803 (2012)
19. Pitz-Paal, R., Botero, N.B., Steinfeld, A.: Heliostat field layout optimization for high-temperature solar thermochemical processing. Solar energy **85**(2), 334–343 (2011)
20. Ramos, A., Ramos, F.: Strategies in tower solar power plant optimization. Solar Energy **86**(9), 2536–2548 (2012)
21. Ramos, A., Ramos, F.: Heliostat blocking and shadowing efficiency in the video-game era. arXiv preprint arXiv:1402.1690 (2014)
22. Salhi, S.: Heuristic Search: The Emerging Science of Problem Solving. Cham: Palgrave Mc Millan (published by Springer) (2017)
23. Sanchez, M., Romero, M.: Methodology for generation of heliostat field layout in central receiver systems based on yearly normalized energy surfaces. Solar Energy **80**(7), 861–874 (2006)
24. Stine, W.B., Geyer, M.: Power from the Sun. Public website: `http://www.powerfromthesun.net/book.html` (Last access in July, 2017) (2001)
25. Tonatiuh Project Website: *Tonatiuh, ray tracing for solar energy.* Public website: `https://github.com/iat-cener/tonatiuh` (Last access in July 2017) (2013)
26. Wang, K., He, Y.L.: Thermodynamic analysis and optimization of a molten salt solar power tower integrated with a recompression supercritical CO_2 brayton cycle based on integrated modeling. Energy Conversion and Management **135**, 336–350 (2017)
27. Wendelin, T., Dobos, A., Lewandowski, A.: SolTrace: A ray-tracing code for complex solar optical systems. Tech. Rep. NREL/TP-5500-59163, NREL (2013)
28. Yao, Y., Hu, Y., Gao, S.: Heliostat field layout methodology in central receiver systems based on efficiency-related distribution. Solar Energy **117**, 114–124 (2015)
29. Yeniay, Ö.: Penalty function methods for constrained optimization with genetic algorithms. Mathematical and Computational Applications **10**(1), 45–56 (2005)
30. Zhang, H., Juchlia, I., Favrat, D., Pelet, X.: Multi-objective thermoeconomic optimisation of the design of heliostat field of solar tower power plants. In: Engineering for Sustainable Energy in Developing Countries (2007)
31. Zhang, H.L., Baeyens, J., Degrève, J., Cacères, G.: Concentrated solar power plants: review and design methodology. Renewable and Sustainable Energy Reviews **22**, 466–481 (2013)
32. Zhang, M., Yang, L., Xu, C., Du, X.: An efficient code to optimize the heliostat field and comparisons between the biomimetic spiral and staggered layout. Renewable Energy **87**, 720–730 (2016)
33. Zhou, Y., Zhao, Y.: Heliostat field layout design for solar tower power plant based on gpu. IFAC Proceedings Volumes **47**(3), 4953–4958 (2014)

A Complementary material

A C++ library with the proposed optimizer and the description of a new heuristic to generate sets of initial solutions can be found at `http://www.hpca.ual.es/~ncalvo/jos17app/appendix.html`