Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Operations Research 33 (2006) 1102–1116
computers & operations research
www.elsevier.com/locate/cor

Efficient hybrid methods for global continuous optimization based on simulated annealing

Kaisa Miettinen[a], Marko M. Mäkelä[b], Heikki Maaranen[b,*]

[a]*Helsinki School of Economics, P.O. Box 1210, FIN-00101 Helsinki, Finland*
[b]*Department of Mathematical Information Technology, University of Jyväskylä, P.O. Box 35, (Agora), FIN-40014, Finland*

Abstract

We introduce several hybrid methods for global continuous optimization. They combine simulated annealing and a local proximal bundle method. Traditionally, the simplest hybrid of a global and a local solver is to call the local solver after the global one, but this does not necessarily produce good results. Besides, using efficient gradient-based local solvers implies that the hybrid can only be applied to differentiable problems. We show several ways how to integrate the local solver as a genuine part of simulated annealing to enable both efficient and reliable solution processes. When using the proximal bundle method as a local solver, it is possible to solve even nondifferentiable problems. The numerical tests show that the hybridization can improve both the efficiency and the reliability of simulated annealing.

Keywords: Global optimization; Metaheuristics; Hybridization; Bundle methods

1. Introduction

In this paper, we study different hybridizations of simulated annealing and a local search method. In *hybridization*, elements of different methods are combined to form new methods. The main interest and motivation in hybridization is to combine elements in a way that they best complement each other. Then it is possible that the new method possesses some desired qualities but not the disadvantages of the original methods. We have developed several hybrids of simulated annealing and a local search. The numerical

* Corresponding author. Tel.: +358 14 260 2765; fax: +358 14 260 2731.
E-mail address: vimaaran@mit.jyu.fi (H. Maaranen).

0305-0548/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2004.09.005

results of the best three hybrids are presented in this paper. We apply the hybrids to a large test suite of problems with continuous variables and show that the new methods can improve both the efficiency and the reliability of simulated annealing.

In recent years, there has been a great deal of interest in developing optimization methods for solving global optimization problems with continuous variables (see, e.g., [1,2] and references therein). This is partly due to the increasing computational capacity of computers, which enables both the use of computationally more expensive search methods and more accurate models of real world problems. The more accurate models are often nonconvex and nondifferentiable. They may also include some noise and the objective function evaluation may involve numerical simulation of some processes. The fact that one cannot necessarily make any assumptions about the form and the shape of the objective function sets high requirements for optimization methods. For example, gradient-based local search methods cannot be applied for nondifferentiable problems, and when applied to multimodal differentiable problems they stagnate to local optima, which may be far from the global ones. For the above-mentioned reasons, more general heuristic methods have been developed.

Metaheuristics are general problem solvers that can successfully solve a wide range of optimization problems. Metaheuristics usually operate directly on objective function values and do not require gradients. They are also able to escape local optima. To improve the rate of convergence, metaheuristics are often hybridized with fast converging local search methods.

There are numerous ways how to hybridize a metaheuristic and a local search method. Hybridization taxonomies are given, for example, in [3–5]. The taxonomies in [3,4] are for hybrids in combinatorial optimization and in [5] the taxonomy considers only hybrids involving genetic algorithms. Nevertheless, all the taxonomies can be generalized for most of the hybrid methods in global optimization. The categories in the taxonomies are not exclusive: a hybrid may have features from several different categories. The benefit of taxonomies is that they help to consider different hybridizations in a systematic manner.

In the taxonomy presented in [3], hybrids are divided into two main categories, sequential and parallel. Parallel hybrids are further divided into synchronous and asynchronous methods. Finally, asynchronous hybrids are divided into homogeneous and heterogeneous methods, which both can be global, partial or functional. In *sequential hybrids*, one algorithm is completed before another algorithm is started, whereas in *parallel hybrids* this is not the case. In *parallel synchronous hybrids* the different algorithms can alter in any predefined manner, but the order of execution is always the same. In *parallel asynchronous hybrids* the different algorithms operate concurrently. This kind of a hybridization is particularly well suited for parallel computing. The parallel asynchronous hybrids are divided into *homogeneous* and *heterogeneous* methods depending on whether the algorithms working in parallel are the same or different, respectively. Furthermore, an asynchronous hybrid is *global*, if all the algorithms operate on the same search space, and *partial* if the search space is divided. If the hybridized algorithms solve different problems, then the cooperation of the algorithms is said to be *functional*. The hybrid taxonomy presented in [4] is an extension of that of [3] and the taxonomy in [5] includes three classes that essentially correspond to (1) sequential and parallel synchronous hybrids, (2) parallel asynchronous hybrids, and (3) hybrids with functional cooperation.

Here, we consider three new parallel synchronous hybrids based on the simulated annealing algorithm presented in [6]. Simulated annealing has been studied extensively during the last two decades. It has its roots in the Metropolis Monte Carlo integration algorithm [7]. But it did not become a popular global search method until it was generalized by Kirkpatrick et al. [8]. Other independent studies were also

done simultaneously (see [9] and references therein). Several positive and negative features of simulated annealing are discussed in [10] including the following most characteristic positive features: simulated annealing can process quite arbitrary objective functions and constraints, it can be easily implemented, and it statistically guarantees finding the global optimum. The main critics of simulated annealing concern the speed of convergence and the fact that the statistical convergence does not provide any guarantee of optimality in practical cases [10]. Some more recent convergence results for simulated annealing are discussed and reviewed in [11,12].

The simulated annealing algorithm by Kirkpatrick et al. was designed for solving combinatorial global optimization problems. Soon variants for continuous variables were also introduced [13,14]. However, simulated annealing algorithms for continuous variables have been proposed by fewer authors than for discrete variables [15]. Some more recent gradient-free optimization methods for continuous variables involving simulated annealing are presented, for example, in [15–22,41]. Some of these methods are hybrids, where simulated annealing is combined with a local search method.

Hybridization with a local search is one approach to speed up simulated annealing (for other approaches, see, e.g., [10] and references therein), but it is not trivial to choose, which local search methods should be used. Fast converging local search methods are often gradient-based. However, simulated annealing is a general problem solver working directly on function values. If a hybridization is made with a gradient-based method, then one of the most positive features of simulated annealing, generality, is lost. Again, if hybridization is made with a non-gradient-based method, the rate of convergence may be very slow.

The recent hybrids involving simulated annealing [15,18,20] are realized in different ways. In [15], a hybrid is proposed, which employs elements of evolutionary algorithms by using a set of working solutions rather than a single solution. In terms of the hybrid taxonomy, it is a homogeneous parallel asynchronous hybrid, where several simulated annealing algorithms work in parallel in the same pool of solutions in global cooperation. Although the algorithm is well suited for parallel computing, the implementation does not make use of it. In [18], simulated annealing is hybridized with a variant of the Nelder–Mead simplex method. It is a parallel synchronous hybrid, where elements of simulated annealing are applied in the acceptance criterion for reflected points in the Nelder–Mead simplex method. In [20], the hybrid approach combines simulated annealing, tabu search and the Nelder–Mead simplex method. It is a hybrid with a two-level structure. On the first level, a parallel synchronous hybrid combines tabu search with the Nelder–Mead simplex method. On the second level, another parallel synchronous hybrid is used, where the first-level hybrid is combined with simulated annealing. An excellent study of 13 different variants of gradient-free simulated annealing is presented in [19], where, among others, simulated annealing is hybridized with a local search and clustering and partitioning techniques. Some of these methods are parallel synchronous and some are parallel asynchronous hybrids.

We have developed 32 different simulated annealing-based hybrids and present here the three most promising methods. As a local search method we use the proximal bundle method [23,24]. The proximal bundle method makes some assumptions about the continuity of the objective function, but does not assume differentiability. For differentiable problems its speed of convergence is comparable to that of gradient-based methods [25]. To put it shortly, the proximal bundle method was selected for its fast convergence and because it can be applied also for nondifferentiable problems.

Sometimes, structures of hybrid methods presented in the literature are so complicated that the methods become difficult to implement. It is also often hard to get an insight into complex methods, and the

adjustment of the parameters becomes difficult. Hence, a simple structure should be preferred to a more complicated one if there is no significant difference in their performance. When testing the 32 hybrid methods numerically, it turned out that the methods with a simple structure also performed better. The difference in the results concerning efficiency and reliability was remarkable when compared to plain simulated annealing. (By reliability we mean the ability of the method to find a global optimum when compared to local methods.) The results were surprising since a simple structure and a good performance were assumed to be conflicting criteria.

The three new methods considered in this paper are parallel synchronous hybrids. Hence, their structure is somewhat similar to [18,20]. Also, the goal in our hybrids of simulated annealing and the proximal bundle method is similar to the goal in [18,20]. We aim to combine the reliability of a global optimization method and the efficiency of a local search method still maintaining the generality of simulated annealing. It is not trivial to try to achieve both of these desirable properties, since they are usually conflicting and the question is which to emphasize in the hybrid. Our hybrids, however, managed to improve both the reliability and the efficiency of simulated annealing.

Despite the similarities to hybrids in [18,20], there are also significant differences. To start with, we use the proximal bundle method instead of the Nelder–Mead simplex method to provide faster convergence. Moreover, we present three different types of hybridization, which gives a better insight to the effects of hybridization. We also shortly discuss some hybridization ideas realized in the other 29 hybrids that did not work so well in practice. In contrast to [20], where the algorithm searches for all the local minima, we are only interested in finding one global minimum.

The main goal of this paper is not to present any single hybrid method. Our goal is to study different parallel synchronized hybrids involving simulated annealing and discuss their performance. For this purpose, any simulated annealing algorithm and any set of test problems could have been chosen. Our interest is to keep this study general. Therefore, we use a simulated annealing algorithm available on the Internet, and as test problems we use a large selection of problem instances from the literature to estimate the average performances of the proposed hybrid methods. To emphasize the generality, we use the same method-specific set of parameters for all the test problems. Moreover, we intentionally suppress details which do not advocate the cause of this general study. For example, the results are presented in a very concise form to emphasize the general trends.

The rest of the paper is organized as follows. We outline the basic features of simulated annealing and the proximal bundle method in Section 2, and Section 3 is devoted to their hybrids. In Section 4, we describe the results of numerical experiments, and discuss the results in Section 5. Finally, the paper is concluded in Section 6.

2. Basic methods

We handle *global optimization problems* of the form

$$\begin{aligned} &\text{minimize} \quad f(\mathbf{x}) \\ &\text{subject to} \quad \mathbf{x}^l \leqslant \mathbf{x} \leqslant \mathbf{x}^u, \end{aligned} \tag{1}$$

where the objective function $f : \mathbf{R}^n \to \mathbf{R}$ is supposed to be locally Lipschitz continuous (see, e.g., [26]) and $\mathbf{x}^l, \mathbf{x}^u \in \mathbf{R}^n$. Note that f is not necessarily differentiable. Next, we present the main elements of the hybrid methods to be introduced.

2.1. Simulated annealing

The basic idea of simulated annealing is to allow the search process to proceed even in a direction where the objective function values deteriorate. In this way, the method tries to escape local optima and find global optima.

Here, we employ the implementation of simulated annealing [27] described in [6]. This implementation has been found to be robust and easy to use as well as applicable to even complex continuous problems. Besides, the implementation is freely available.

The basic idea of the algorithm is to combine local and global search in order to get more accurate global solutions. To start with, the probability of accepting worse solutions is reasonably high and new candidate solutions are selected from a relatively large set. During the search process, the probability is decreased and the set is reduced. In this way, the search first emphasizes a global and after that a local search.

Suppose that we have the current iteration point $\mathbf{x} \in \mathbf{R}^n$. The next candidate solution $\mathbf{y}$ is generated by varying one component $i \in \{1, \ldots, n\}$ of $\mathbf{x}$ at a time, in other words,

$$y_i = x_i + q \cdot d_i, \tag{2}$$

where q is a uniformly distributed random number from $[-1, 1]$ and $\mathbf{d} \in \mathbf{R}^n$ is a search direction. The point $\mathbf{y}$ is accepted as the next iteration point if it is better than the current iteration point, in other words, $f(\mathbf{y}) < f(\mathbf{x})$ or if it is worse than the current iteration point but a so-called Metropolis criterion is valid. In other words,

$$\mathrm{e}^{(f(\mathbf{x})-f(\mathbf{y}))/t} > p,$$

where p is a uniformly distributed random number from [0,1]. Here, the parameter $t > 0$ is a so-called temperature and it is decreased during the algorithm.

The basic formulations of simulated annealing do not specifically consider the choice of $\mathbf{d}$. In the implementation used, the length of the search direction $\mathbf{d}$ is adjusted componentwise so that about a half of the candidate solutions are accepted. The goal of this so-called *adaptive adjustment* is to sample the function values widely. If more than 60% of the candidates are accepted, then the length of $\mathbf{d}$ is extended. For a given temperature, this increases the number of rejections and decreases the percentage of acceptances. On the other hand, if the ratio of acceptance is less than 40%, then the length of $\mathbf{d}$ is correspondingly decreased [6].

The concise algorithm for simulated annealing is the following:

SET initial parameters,
starting point $\mathbf{x}^l \leqslant \mathbf{x} \leqslant \mathbf{x}^u$,
search direction $\mathbf{d}$,
initial temperature $t > 0$,
temperature reduction factor $r_t \in (0, 1)$,
final accuracy tolerance $\varepsilon > 0$,
number of function values used in termination criteria N_ε,
number of cycles before search direction adjustment N_s,
number of iterations before temperature reduction N_t.

```
CALCULATE f_x = f(x),
          x* = x,
          f* = f_x.
DO UNTIL convergence
          DO N_t times
             DO N_s times
                DO i = 1, ..., n,
                   y_i = x_i + q · d_i (q is a uniform random number on [−1, 1]).
                   CALCULATE f_y = f(y),
                   IF f_y < f_x THEN
                      x = y, f_x = f_y.
                      IF f_y < f* THEN
                         x* = y, f* = f_y.
                      END IF
                   ELSE
                      IF e^((f_x − f_y)/t) > p THEN (p is uniform random number on [0,1])
                         x = y, f_x = f_y,
                      END IF
                   END IF
                END DO
             END DO
             ADJUST d such that half of all candidate solutions are accepted
          END DO
          IF change in f* < ε during the last N_ε iterations AND |f_x − f_y| < ε THEN
             REPORT x*, f*
             STOP
          ELSE
                x = x*, f_x = f*,
                t = r_t t.
          END IF
CONTINUE
```

Note that varying one component of the current iteration point at a time in (2) is not necessarily efficient in handling real-life problems where variables may be highly correlated. In such cases, it may be useful, for example, to choose the component randomly or to vary several components together. However, the algorithm presented is sufficient for our purposes as a general solver for hybrid developments and as a solid reference for performance comparisons.

2.2. Proximal bundle method

The proximal bundle method [23,24] is a local optimization method capable of solving even nondifferentiable problems. The central idea of bundle methods is to generalize efficient gradient-based methods to locally Lipschitz continuous functions, which do not necessarily have gradients in a classical sense. This is realized via bundling subgradients [26] that generalize gradient information.

Next we shortly present the basic idea of the proximal bundle method for local nondifferentiable optimization. Suppose that we have the current iteration point $\mathbf{x} \in \mathbf{R}^n$. The next candidate solution is generated by $\mathbf{y} = \mathbf{x} + \lambda\mathbf{d}$, where the search direction $\mathbf{d} \in \mathbf{R}^n$ is generated by solving a quadratic subproblem (see appendix) and $\lambda > 0$ is the step length into the search direction. In the subproblem, the original objective function is approximated by a polyhedral function containing a bundle of subgradient information.

The method is said to take a so-called *serious step*, in other words, the point $\mathbf{y}$ is accepted as the next iteration point if it is clearly better than the current one, that is

$$f(\mathbf{y}) < f(\mathbf{x}) - \delta\lambda, \tag{3}$$

where the tolerance $\delta > 0$ can be controlled by the user. Otherwise, a so-called *null step* occurs. In other words, the current iteration point $\mathbf{x}$ remains but a new subgradient is added into the bundle of subgradients in order to improve the polyhedral approximation of the objective function. For more details, we refer to appendix and [23,24].

If we compare the proximal bundle method to the derivative-free methods like the Nelder–Mead simplex method, the convergence is much faster. As a matter of fact, the convergence speed of bundle methods is comparable to that of the conjugate gradient method [25]. Besides, the proximal bundle method can efficiently solve relatively large problems [28]. On the other hand, the proximal bundle method does not necessitate the differentiability of the functions involved as gradient-based methods do. These two properties of the proximal bundle method are its advantages when compared to the local solvers generally used in hybrid methods (as discussed in the introduction).

3. New hybrids

A traditional way to hybridize global and local optimizers is to start local optimization from the final solution of the global solver. This is a very straightforward sequential hybridization by which the accuracy of the solution can often be improved. Unfortunately, the local phase increases the computational costs that usually are quite high already in the plain global optimization. If the costs are to be decreased, it is possible to stop the global phase somewhat earlier and start the local phase in order to guarantee optimality. The problem here is when to make the switch. If the global phase is terminated too early, the probability of finding the global optimum is decreased. On the other hand, continuing it for too long causes extra computational costs.

In more genuine hybrids, different optimizers are integrated so that their roles are mixed. In this paper, we integrate simulated annealing and the proximal bundle method. The original idea was to make a parallel synchronized hybrid where the idea of simulated annealing is embedded into the proximal bundle method. To be more specific, a serious step is to be taken in the proximal bundle method if (3) is valid, or if the Metropolis criterion is valid. In this case, the latter criterion is

$$\mathrm{e}^{(f(\mathbf{x})-f(\mathbf{y}))/t_{\mathrm{loc}}} > p,$$

where t_{loc} is an iteration-dependent temperature, and it approaches zero when iterations increase. In this way, the proximal bundle method can be augmented to be able to escape local optima, and the hybrid can be called a *biased proximal bundle method*. Unfortunately, computational experiments did not support this

idea because the search directions generated by the proximal bundle method were not versatile enough. Thus, the global aspects of the method were inadequate.

In the next parallel synchronized hybrid, the idea is to use the proximal bundle method as a part of simulated annealing and not vice versa. This means that whenever a candidate solution is accepted in simulated annealing, the proximal bundle method is applied starting from that point. This increases the local accuracy without compromising the global search. The original simulated annealing is here modified so that the adaptive adjustment is ignored. Thus, instead of using (2), we used the formula $y_i = x_i^l + q(x_i^u - x_i^l)$, where $q \in [0,1]$ is a uniform random number. We call this approach by the name *hybrid A*.

The idea of the third hybridization is to make a two-level parallel synchronized hybrid by combining the basic principles of the two hybrids, biased proximal bundle method and hybrid *A*, described above. In other words, the proximal bundle method used in hybrid *A* is now replaced by the biased proximal bundle method. It is important to notice that this *hybrid B* contains two nested cooling strategies, which means that we have two temperature parameters to be updated independently. In this way, we can emphasize the global search during the local search.

In the last parallel synchronized hybrid, *hybrid C*, the motivation is to decrease the computational cost of hybrid *A*. This is realized by starting the local search with a relatively low accuracy and increasing it as the temperature is decreased during the optimization process. This means that the stopping parameter of the proximal bundle method (see appendix) is $\varepsilon_{\mathrm{loc}} = \alpha t$, where t is the iteration-dependent temperature and $\alpha > 0$ is a scaling parameter.

Even though the hybrids *A*, *B*, and *C* are very simple to implement, their performances are surprisingly good when compared to simulated annealing. The numerical experiments supporting this conclusion are described in more detail in the next section.

4. Numerical experiments

Numerical experiments were carried out to test the performance of the different hybrids. As benchmark methods we used simulated annealing with two sets of parameters. A number of 38 test problems of different types were selected from the literature. The names of the test problems are given in Table 1 along with the number of variables, number of local minima, box-constraints, and references. In column '# of minima' in Table 1, 'N/A' stands for the case where the number of local minima for the problem could not be found in the literature. The box-constraints are given as intervals for each variable unless all variables are restricted to the same interval, in which case only one interval is given.

All the hybrids were implemented in Fortran 77 and the test runs were performed on an HP9000/J5600 computer. The simulated annealing implementation used is available at [27]. The proximal bundle implementation is PBNCGC described in [24], which uses the quadratic solver QPDF4 presented in [32].

We call the two benchmark methods by the names SA I and SA II. They differ only in respect of the parameter value N_t (the number of cycles before temperature reduction). For SA I we used the value $N_t = \max\{100, 5n\}$ recommended in [6], and for SA II we employed the value $N_t = 5$ used in [27]. Other values for N_t were also tested, but the values recommended in [6] and [27] proved to be good for the simulated annealing algorithm. In general, the value for N_t used in [6] emphasizes robustness at the expense of computational cost, and the reverse is true for the value used in [27].

Table 1
Summary of the test problems

#	Name	Dimensions (n)	# of minima	Box constraints	Ref.
1–3	Michalewicz	2, 5, 10	N/A	$[0, \pi]$	[29]
4–7	Schwefel	6, 10, 20, 50	N/A	$[-500, 500]$	[29]
8	Branin rcos	2	3	$[-5, 10]$ $[0, 15]$	[29]
9–13	Griewangk	2, 6, 10, 20, 50	10^3 for $n = 10$	$[-600, 600]$	[29]
14–18	Ackley's path	2, 6, 10, 20, 30	N/A	$[-32.768, 32.768]$	[29]
19	Easom	2	1	$[-100, 100]$	[29]
20	Levy	4	71.000	$[-10, 10]$	[30]
21–23	Levy	5, 6, 7	10^5, 10^6, 10^8	$[-5, 5]$	[30]
24	Hansen	2	760	$[-10, 10]$	[30]
25–28	Funtion 10	3, 4, 10, 20	2^n	$[-20, 20]$	[31]
29–31	Shekel	4, 4, 4	5, 7, 10	$[0, 10]$	[30]
32	Rosenbrock	2	1	$[-2, 4]$ $[-2, 2]$	[30]
33	6-hump camel back	2	N/A	$[-50, 50]$	[30]
34	No name	2	N/A	$[-2, 2]$	[a]
35	Problem 257	4	N /A	$[-10^6, 10^6]$	[21]
36–38	Rastrigin	2, 4, 6	50 for $n = 2$	$[-5.12, 5.12]$	[29]

[a] $f(\mathbf{x}) = x_1 \sin(x_1 x)_2 \cos(x_2)$.

Table 2
Parameters

Parameters	SA I	SA II	Hybrids
$t/r_t/\varepsilon$	$5.0/0.85/10^{-6}$	$5.0/0.85/10^{-6}$	$5.0/0.85/10^{-6}$
$N_\varepsilon/N_S/N_t$	4/20/max{100,5n}	4/20/**5**	4/**10**/**1**
t_{loc}/r_{loc} (hybrid B)	—	—	**5**/**0.75**

The parameter values used for SA I, SA II, and for the hybrid methods A, B, and C are given in Table 2. The values differing from those of SA I are written in boldface. As the starting point for the search we used $\mathbf{x}^l$. The initial search direction was $\mathbf{d} = (1, \ldots, 1)^T$ for SA I and SA II. As a reminder, t is the initial temperature, r_t is the temperature reduction factor, ε is the final accuracy tolerance, N_ε is the number of function values used in termination criteria, N_S is the number of cycles before search direction adjustment, and N_t is the number of cycles before temperature reduction. The local search parameters t_{loc} and r_{loc} are used only in hybrid B. The parameter $r_{loc} \in (0,1)$ is a user-defined coefficient that is used to control how fast the temperature t_{loc} is decreased in the biased proximal bundle method. This was done by multiplying t_{loc} with r_{loc} after each iteration. In addition, in hybrid C, we set the scaling parameter $\alpha = 10^{-3}$. This choice is based on the numerical experiments with the 38 test problems. Let us mention that the performance of the method is not sensitive to the choice of this parameter.

In Table 3, we present the summarized results of the numerical tests. Each of the 38 test problems was solved ten times using SA I and SA II as well as using hybrids A, B, and C. The row 'best' indicates

Table 3
Summary of numerical experiments

	SA I	SA II	Hybrid *A*	Hybrid *B*	Hybrid *C*
Best	2.6%	7.1%	28.2%	13.9%	44.2%
Fail-opt.	26.1%	41.6%	11.3%	18.4%	9.5%
Fail-acc.	5.3%	6.3%	7.1%	5.0%	7.6%
Max iter.	—	—	—	5.8%	—
Fcn evals	3 089 906	104 007	19 253	35 121	12 284

the percentage of the test runs where each of the methods found the global optimum[1] within the desired accuracy ($\varepsilon = 10^{-2}$) and used least function evaluations when compared to the other methods. Moreover, another tolerance for the objective function value was used to distinguish when the methods failed to find the global optimum and when they merely failed in accuracy. In Table 3, the percentage of problems where the methods failed to find the global optimum within a tolerance 10^{-1} is denoted by 'fail-opt.'. The row 'fail-acc.' shows the percentage of problems where the accuracy was between 10^{-2} and 10^{-1}. The row 'max iter.' indicates the percentage of problems where the stopping criterion of simulated annealing could not stop the search. Finally, 'fcn evals' is the average number of function evaluations, that is, the total number of function evaluations for all the 38 test problems divided by the number of problems and the number of test runs. Note that, if max iter. was reached, the number of function evaluations was not taken into account when calculating the average of function evaluations.

Fig. 1 illustrates the average numbers of objective function evaluations that the method SA II and the hybrids *A*, *B*, and *C* used for different problems. The method SA I is not illustrated since its number of function evaluations was of a much larger order. Again, if max iter. was reached, the number of function evaluations was not plotted in Fig. 1. For the plotting, the problems were divided into problem sets 1 and 2. The set 1 includes the problems, where the benchmark method SA II used less than 100 000 function evaluations (the upper plot in Fig. 1) and the rest of the problems belong to set 2 (the lower plot in Fig. 1).

Based on the numerical experiments we can conclude that the hybrid *C* is the most efficient and the most reliable among the methods tested. The hybrid *A* is almost equally reliable but not quite as efficient. In Fig. 1 we see that these two methods dominated others in nearly all the problems. Even though the computational costs of simulated annealing could be dramatically decreased by tuning the parameters (in SA I and SA II), the hybrids are superior in this aspect. This means that hybridization did not increase the computational costs but vice versa. What is most important, the computational savings did not imply weaker performance in terms of reliability, opposed to what one could expect. These results speak strongly for the hybridization. An extra effort for implementation, which sometimes is a downside of hybrid methods, is not an issue for hybrids *A* and *C*. The hybridizations are straightforward, and there is a large number of freely available local search methods, which can be used in hybridization with a very low effort. In summary, the hybrid methods presented in this paper do not need much extra effort in implementation, nevertheless, they are both more efficient and more reliable than simulated annealing using adaptive adjustment.

[1] By a global optimum we here mean the value given in the literature or, if this was not available, the best value obtained in our previous tests.

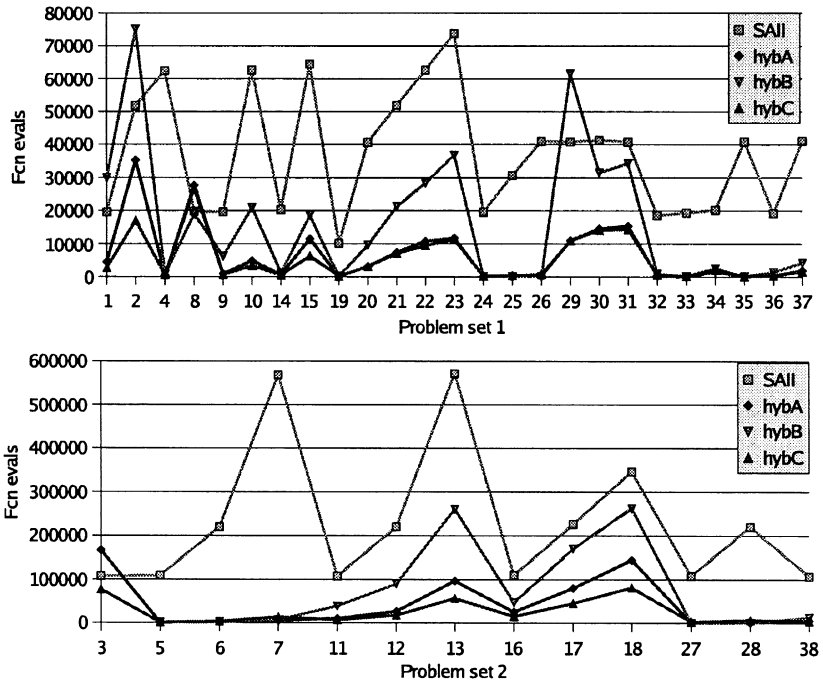

Fig. 1. Number of average function evaluations for SA II and hybrids A, B, and C.

5. Discussion

The three hybrid methods introduced are the results of extensive development work supported by numerical experiments. In all, 32 different variants of hybridization have been treated. It was clear in the very beginning that hybridizing simulated annealing with a local solver is sensible only if the adaptive adjustment of the search direction is ignored. This is explained by the fact that the adjustment has been developed to represent local search in simulated annealing. This is evidenced by the computational experiments of applying traditional hybridization, that is, local search after simulated annealing where the accuracy of the solution could not be improved. On the other hand, the additional local phase could not improve the reliability of finding global solutions.

One of the ideas treated was applying local search to every candidate solution in simulated annealing before testing its acceptance. The hybrids A, B, and C were also applied so that the local search was not used in early iterations of the simulated annealing in order to decrease computational costs. Even though these methods were superior to simulated annealing, they did not improve the performance compared to the simpler hybrids A, B, and C. This indicates that seemingly good ideas do not necessarily work well in practice. Hence, numerical tests are always required.

It is in order to note that the local optimization method, that is, the proximal bundle method requires the evaluation of subgradients, which increases the efficiency to a significant extent, but which also increases computational costs. When the number of variables is high the balance between efficiency and computational costs is likely to become unfavorable for a hybrid applying the proximal bundle method. We should point out that any other local optimization method can be used in hybrids A and C. In case of large-dimensional problems, gradient-free methods can be applied.

Let us point out that the test suite commonly used in the literature [18,33–39] includes mainly relatively easy problems, that is, problems that almost all the reported methods solve using only a few hundred function evaluations. We have selected a more challenging test suite, since, in our opinion, there is a need for methods capable of solving also difficult problems and the test problems should reflect that ambition. For example, the methods in [18,34–36,39] stumble more or less in solving the Shekel family of functions, whereas none of our hybrids failed in those problems. This indicates that the methods in the literature are often designed or tuned for relatively easy problems and, therefore, provide no good comparison for our purposes. Due to the choice of more difficult, but less common, test problems, the comparison to existing methods becomes complicated. However, we have applied an evolutionary algorithm for mostly the same test suite in [40]. With the parameter values used in [40], the evolutionary algorithm was more reliable than SA II using approximately the same number of function evaluations. However, the here proposed hybrid methods are more efficient than the evolutionary algorithm used in [40] and their reliability is very competitive.

A fair comparison between inherently different methods using difficult test problems is a highly challenging task, not least because of the different parameters that need to be adjusted separately. Hence, the comparison is an interesting subject for future research. However, in this paper, we have concentrated on the improvements on simulated annealing that can be achieved via hybridization with a local search method. Therefore, basic simulated annealing was a natural selection as a benchmark method.

6. Conclusions

We have introduced new hybrid methods for global continuous optimization. They are based on simulated annealing and a local proximal bundle method. With extensive numerical experiments we have demonstrated the efficiency and the reliability of the hybrids. It is notable that both efficiency and reliability of simulated annealing could be improved significantly. Moreover, this work provides important information about the performance of different parallel synchronous hybrids involving simulated annealing.

Acknowledgements

This research was supported by the Academy of Finland, Grant #102020 as well as the National Technology Agency of Finland.

Appendix. More detailed description of proximal bundle method

Next we shortly present the proximal bundle method for local nondifferentiable optimization. For more details, we refer to [23,24].

We suppose that for each $\mathbf{x} \in \mathbf{R}^n$ we can evaluate a subgradient $\xi \in \partial f(\mathbf{x})$, where

$$\partial f(\mathbf{x}) = \operatorname{conv}\left\{ \lim_{i\to\infty} \nabla f(\mathbf{x}^i) \mid \mathbf{x}^i \to \mathbf{x},\ \nabla f(\mathbf{x}^i) \text{ exists and converges} \right\} \tag{4}$$

is the subdifferential of the function f in the sense of [26] and "conv'' denotes the convex hull of a set.

Suppose that in addition to the current iteration point $\mathbf{x}_k$ we have some trial points $\mathbf{y}_j \in \mathbf{R}^n$ (from past iterations) and subgradients $\xi_j \in \partial f(\mathbf{y}_j)$ for $j \in J^k$, where the index set $J^k \neq \emptyset$ is assumed to be a subset of $\{1, \ldots, k\}$.

The idea of bundle methods is to approximate the objective function f from below. For that purpose we define a convex polyhedral approximation by

$$\hat{f}^k(\mathbf{x}) = \max_{j \in J^k} \{f(\mathbf{x}_k) + \xi_j^{\mathrm{T}}(\mathbf{x} - \mathbf{x}_k) - \beta_j^k\}, \tag{5}$$

with the subgradient locality measure

$$\beta_j^k = \max\{|f(\mathbf{x}_k) - f(\mathbf{y}_j) - \xi_j^{\mathrm{T}}(\mathbf{x}_k - \mathbf{y}_j)|, \gamma\|\mathbf{x}_k - \mathbf{y}_j\|^2\}, \tag{6}$$

where $\gamma \geqslant 0$ is a distance measure parameter ($\gamma = 0$ if f is convex). Now we can form an approximation to problem (1) by

$$\begin{array}{ll} \text{minimize} & \hat{f}^k(\mathbf{x}_k + \mathbf{d}) + \frac{1}{2}u_k\|\mathbf{d}\|^2 \\ \text{subject to} & \mathbf{x}^l \leqslant \mathbf{x}_k + \mathbf{d} \leqslant \mathbf{x}^u, \end{array} \tag{7}$$

where $u_k > 0$ is some weighting parameter to improve the convergence rate and to accumulate some second order information. The role of the regularizing quadratic penalty term $\frac{1}{2}u_k\|\mathbf{d}\|^2$ is to guarantee the existence of the solution $\mathbf{d}_k$ and to keep the approximation local enough.

Now we have calculated the search direction $\mathbf{d}_k$. Next we consider the problem of determining the step size in that direction. We assume that $m_{\mathrm{L}} \in (0, \frac{1}{2})$, $m_{\mathrm{R}} \in (m_{\mathrm{L}}, 1)$ and $\bar{\lambda} \in (0, 1]$ are fixed line search parameters. First we search for the largest number $\lambda_{\mathrm{L}}^k \in [\bar{\lambda}, 1]$ such that

$$f(\mathbf{x}_k + \lambda_{\mathrm{L}}^k \mathbf{d}_k) \leqslant f(\mathbf{x}_k) + m_{\mathrm{L}}\lambda_{\mathrm{L}}^k v_k, \tag{8}$$

where $v_k = \hat{f}^k(\mathbf{x}_k + \lambda_{\mathrm{L}}^k \mathbf{d}_k) - f(\mathbf{x}_k)$ is a predicted descent. If such a parameter λ_{L}^k exists, we take a so-called long serious step

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda_{\mathrm{L}}^k \mathbf{d}_k \quad \text{and} \quad \mathbf{y}_{k+1} = \mathbf{x}_{k+1}. \tag{9}$$

Otherwise, if (8) is valid but $0 < \lambda_{\mathrm{L}}^k < \bar{\lambda}$, then a short serious step

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda_{\mathrm{L}}^k \mathbf{d}_k \quad \text{and} \quad \mathbf{y}_{k+1} = \mathbf{x}_k + \lambda_{\mathrm{R}}^k \mathbf{d}_k \tag{10}$$

is taken. Finally, if $\lambda_{\mathrm{L}}^k = 0$ we take a null step

$$\mathbf{x}_{k+1} = \mathbf{x}_k \quad \text{and} \quad \mathbf{y}_{k+1} = \mathbf{x}_k + \lambda_{\mathrm{R}}^k \mathbf{d}_k, \tag{11}$$

where $\lambda_{\mathrm{R}}^k > \lambda_{\mathrm{L}}^k$ is such that

$$-\beta_{k+1}^{k+1} + \xi_{k+1}^{\mathrm{T}} \mathbf{d}_k \geqslant m_{\mathrm{R}} v_k. \tag{12}$$

In numerical calculations, we use the line search algorithm by [24] based on the above strategies. For updating the weighting parameter u_k we use the safeguarded quadratic interpolation algorithm presented in [23].

The algorithm is terminated if

$$|v_k| \leqslant \varepsilon_{\mathrm{loc}}, \tag{13}$$

where $\varepsilon_{\mathrm{loc}} > 0$ is a final accuracy tolerance supplied by the user.

As far as the parameter values are concerned, we set $\gamma = 1.5$, $m_L = 0.01$, $m_R = 0.5$, $\bar{\lambda} = 0.1$ and $\varepsilon_{loc} = 10^{-6}$.

References

[1] Floudas CA, Pardalos PM, editors. Recent advances in global optimization. Princeton, NJ: Princeton University Press; 1992.
[2] Horst R, Pardalos PM, editors. Handbook of global optimization. Dordrecht: Kluwer Academic Publishers; 1995.
[3] Preux Ph, Talbi E-G. Towards hybrid evolutionary algorithms. International Transactions in Operational Research 1999;6(6):557–70.
[4] Talbi E-G. Taxonomy of hybrid metaheuristics. Journal of Heuristics 2002;8(5):541–64.
[5] Yen J, Liao JC, Lee B, Randolph D. A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method. IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics 1998;28(2):173–91.
[6] Goffe WL, Ferrier GD, Rogers J. Global optimization and statistical functions with simulated annealing. Journal of Econometrics 1994;60(1–2):65–99.
[7] Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E. Equation of state calculation by fast computing machines. Journal of Chemical Physics 1953;21(6):1087–92.
[8] Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. Science 1983;220(4598):671–80.
[9] Ingber L. Adaptive simulated annealing (ASA): lessons learned. Control and Cybernetics 1996;25(1):33–54.
[10] Ingber L. Simulated annealing: practice versus theory. Mathematical and Computer Modelling 1993;18(11):29–57.
[11] Locatelli M. Simulated annealing algorithms for continuous global optimization: convergence conditions. Journal of Optimization Theory and Applications 2000;104(1):121–33.
[12] Yang RL. Convergence of the simulated annealing algorithm for continuous global optimization. Journal of Optimization Theory and Applications 2000;104(3):691–716.
[13] Aluffi-Pentini F, Parisi V, Zirilli F. Global optimization and stochastic differential equations. Journal of Optimization Theory and Applications 1985;47:1–16.
[14] Vanderbilt D, Louie SG. A Monte Carlo simulated annealing approach to optimization over continuous variables. Journal of Computational Physics 1984;56:259–71.
[15] Ali MM, Törn A, Viitanen S. A direct search variant of the simulated annealing algorithm for optimization involving continuous variables. Computers & Operations Research 2002;29(1):87–102.
[16] Bilbro G, Snyder W. Optimization of functions with many minima. IEEE Transactions on Systems, Man, and Cybernetics 1991;21(4):840–9.
[17] Decker A, Aarts E. Global optimization and simulated annealing. Mathematical Programming 1991;50(3):367–93.
[18] Hedar A-L, Fukushima M. Hybrid simulated annealing and direct search method for nonlinear unconstrained global optimization. Optimization Methods and Software 2002;17(5):891–912.
[19] Özdamar L, Demirhan M. Experiments with new stochastic global optimization search techniques. Computers & Operations Research 2000;27(9):841–65.
[20] Salhi S, Queen NM. A hybrid algorithm for identifying global and local minima when optimizing functions with many minima. European Journal of Operational Research 2004;155(1):51–67.
[21] Schittkowski K. More test examples for nonlinear programming codes. Berlin: Springer; 1987.
[22] Wang PP, Chen D-S. Continuous optimization by a variant of simulated annealing. Computational Optimization and Applications 1996;6(1):59–71.
[23] Kiwiel KC. Proximity control in bundle methods for convex non-differentiable optimization. Mathematical Programming 1990;46(1):105–22.
[24] Mäkelä MM, Neittaanmäki P. Nonsmooth optimization: analysis and algorithms with applications to optimal control. Singapore: World Scientific Publishing Co.; 1992.
[25] Lemaréchal C. Nondifferentiable optimization. In: Nemhauser GL, Rinnooy Kan AHG, Todd MJ., editors. Optimization. Amsterdam: North-Holland; 1989. p. 529–72.
[26] Clarke FH. Optimization and nonsmooth analysis. New York: Wiley; 1983.
[27] SIMANN—Fortran simulated annealing code used in [6]. Available at http://wueconb.wustl.edu/~goffe/, June 2004.

[28] Haarala M, Miettinen K, Mäkelä MM. New limited memory bundle method for large-scale nonsmooth optimization. Optimization Methods and Software 2004;19(6):673–92.
[29] Genetic and evolutionary algorithm toolbox for use with MatLab, http://www.geatbx.com/, June 2004.
[30] Test problems for global optimization, http://www.imm.dtu.dk/~km/GlobOpt/testex/, June 2004.
[31] Trafalis B, Kasap S. A novel metaheuristics approach for continuous global optimization. Journal of Global Optimization 2002;23(2):171–90.
[32] Kiwiel KC. A method for solving certain quadratic programming problems arising in nonsmooth optimization. IMA Journal of Numerical Analysis 1986;6:137–52.
[33] Battiti R, Tecchiolli G. The continuous reactive tabu search: blending combinatorial optimization and stochastic search for global optimization. Annals of Operations Research 1996;63:153–88.
[34] Bessaou M, Siarry P. A genetic algorithm with real-value coding to optimize multimodal continuous functions. Structural and Multidisciplinary Optimization 2001;23(1):63–74.
[35] Chelouah R, Siarry P. Tabu search applied to global optimization. European Journal of Operational Research 2000;123(2):256–70.
[36] Chelouah R, Siarry P. A hybrid method combining continuous tabu search and Nelder-Mead simplex algorithms for the global optimization of multiminima functions. European Journal of Operational Research, in press.
[37] Ji M, Tang H. Global optimizations and tabu search based on memory. Applied Mathematics and Computation 2004;159(2):449–57.
[38] Siarry P, Berthiau G. Fitting of tabu search to optimize functions of continuous variables. International Journal for Numerical Methods in Engineering 1997;40(13):2449–57.
[39] Siarry P, Berthiau G, Durdin F, Haussy J. Enhanced simulated annealing for globally minimizing functions of many-continuous variables. ACM Transactions on Mathematical Software 1997;23(2):209–28.
[40] Maaranen H, Miettinen K, Mäkelä MM. Quasi random initial population for genetic algorithms. Computers & Mathematics with Applications 2004;47(12):1885–95.
[41] Ali MM, Storey C. Aspiration based simulated annealing algorithm. Journal of Global Optimization 1997;11(2):181–91.