J Glob Optim (2016) 66:511–534
DOI 10.1007/s10898-016-0417-5

Global versus local search: the impact of population sizes on evolutionary algorithm performance

Thomas Weise[1] · Yuezhong Wu[1] · Raymond Chiong[2] · Ke Tang[1] · Jörg Lässig[3]

Received: 3 March 2015 / Accepted: 12 February 2016 / Published online: 23 February 2016

Abstract In the field of Evolutionary Computation, a common myth that "An Evolutionary Algorithm (EA) will outperform a local search algorithm, given enough runtime and a large-enough population" exists. We believe that this is not necessarily true and challenge the statement with several simple considerations. We then investigate the population size parameter of EAs, as this is the element in the above claim that can be controlled. We conduct a related work study, which substantiates the assumption that there should be an optimal setting for the population size at which a specific EA would perform best on a given problem instance and computational budget. Subsequently, we carry out a large-scale experimental study on 68 instances of the Traveling Salesman Problem with static population sizes that are powers of two between $(1 + 2)$ and $(262144 + 524288)$ EAs as well as with adaptive population sizes. We find that analyzing the performance of the different setups over runtime supports our point of view and the existence of optimal finite population size settings.

✉ Thomas Weise
tweise@ustc.edu.cn

Yuezhong Wu
yuezhong@mail.ustc.edu.cn

Raymond Chiong
Raymond.Chiong@newcastle.edu.au

Ke Tang
ketang@ustc.edu.cn

Jörg Lässig
jlaessig@hszg.de

1 Joint USTC-Birmingham Research Institute in Intelligent Computation and Its Applications (UBRI), School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, Anhui, China

2 School of Design, Communication and Information Technology, Faculty of Science and Information Technology, The University of Newcastle, Callaghan, NSW 2308, Australia

3 Faculty of Electrical Engineering and Computer Science, Hochschule Zittau/Görlitz, 02826 Görlitz, Germany

Keywords Population size · Traveling salesman problem · Experimentation · Statistics · Evolutionary computation

1 Introduction

Evolutionary Algorithms (EAs) [5,17,26,84,85] are population-based global optimization algorithms. Despite the fact that much research has been done showing that larger populations are not always useful, it is still a common belief in the Evolutionary Computation (EC) community that

> An EA will outperform a local search algorithm, given enough runtime and a large-enough population.

In this article, we aim to

1. challenge this statement from several angles, and
2. analyze the impact of the population size parameter on the performance of EAs, both with a literature review and an experimental study to support our arguments.

We will first point out some simple issues with the statement above in Sect. 2, and then review the literature related to population size impact in Sect. 3. We support our arguments with a comprehensive experimental study using the Traveling Salesman Problem (TSP) [3, 29,49,52] as a test bed in Sect. 4. Our goal is to provide evidence showing that the two conditions *enough runtime* and *large-enough populations* are *not* sufficient to let an EA outperform another EA with a smaller population, let alone a good local search algorithm. Instead, we show that there are static population size settings that are optimal in terms of performance for a given computational budget. In Sect. 5, we investigate whether (simple) adaptive population sizing schemes can change this situation qualitatively. We find that the behavior of the adaptation schemes we tested provides further support for our arguments. Finally, in Sect. 6, we summarize our findings and give an outlook on future work on this topic.

2 Basic considerations

2.1 What does "outperform" mean?

A first issue with the initial statement emerges from the fact that there is no clear definition of what "outperform" means. Often, "outperform" is interpreted as "finding better solutions". As pointed out in [90], this approach is not sufficient as it ignores the runtime requirement. Consider a situation where a local search finds a solution that is 0.001 % worse than the global optimum within one second, while an EA needs one month to find such a solution and *later* discovers the global optimum. Only if runtime does not matter, it can be stated that the EA has outperformed the local search. In practical scenarios, runtime always matters.

Exact deterministic algorithms, random walks, or random sampling will, too, sooner or later find the global optimum. If finding the global optimum is indeed the criterion for whether an optimization algorithm is good or not, it must do so faster than these algorithms. This invalidates assumptions about an infinite population and infinite runtime.

2.2 Better solutions if enough runtime

One premise often encountered in the field of optimization is that a global optimization algorithm, such as an EA or a Swarm Intelligence [12,79] method, can find better solutions than a local search algorithm if enough runtime is granted. It is indeed true that several global optimization methods can find optimal or close-to-optimal solutions if given enough time [9]. However, in many problems, suitable local search algorithms can find the global optimum, too. If they can do so faster than an EA, it does not matter how much runtime is granted, the EA cannot be considered as better than the local search. A good example of this is the TSP, where approaches based on the Lin–Kernighan heuristic [54,97], a local search, are currently considered to be the best-performing algorithms and can solve many TSP instances to optimality [34].

2.3 Outperform if large-enough populations

It is further believed that large populations in EAs can prevent premature convergence, i.e., getting stuck in a local optimum. In other words, if the population is large enough, an EA will find the global optimum eventually. Then again, the runtime needed for one generation of an EA is proportional to the population size. If the population of the EA is so large that the computational budget of the optimization process is exhausted during the first generation, the EA will behave like a random sampling algorithm [61]: It will only generate the random (initial) solutions of the first generation and then terminate. Random sampling algorithms are not considered to be good optimization methods and usually will not outperform even trivial local search algorithms. Besides memory, runtime constraints therefore put hard limits on the population size. In other words, it may not be possible to simply increase the population size of an EA until it finds better solutions than a local search method.

2.4 Finding the global optimum

Even if an EA is granted long runtime and a large population, it is not guaranteed that it will find the global optimum: It may converge prematurely [86,88].

A search operation is *complete* [80,85,88] if it can produce any possible solutions from any other possible solutions. If the unary (mutation) operation of the EA is complete, the EA will eventually escape the basin of attraction of the local optimum trapping it and find the global optimum. However, mutation operators necessarily have a bias to create solutions that are similar to their parents. The production of completely different solutions is much less likely. Without this bias, mutations would be the same as random sampling steps. Random sampling creates any possible solutions with the same probability. In a prematurely converged EA, escaping the current local optimum with mutation may have a smaller probability due to this bias. It could potentially take much longer to find the optimum with that EA than finding it with random sampling.

The use of binary (crossover) operations is one of the elements distinguishing EAs from local search methods. Crossover provably has a tremendous positive impact on the convergence speed of the search [19] in some (but not all [2]) problems. Crossover operators usually produce new solutions that either contain parts of the input (parent) genotypes or are sampled from a subspace defined by these parents. Such crossover operators are not complete since, for two given parent solutions, there are points in the search space they cannot produce. In a prematurely converged EA, the presence of crossover operators and the allocation of trials to them may decrease the probability of escaping the local optimum.

The global search ability of an EA therefore does not guarantee that it will reach a better solution than local search. In [86,88], we have provided some detailed analysis on several problematic aspects that may cause premature convergence, as well as potential remedies for them.

2.5 Representation design

It is well-known that the performance of an EA can be significantly improved by choosing a good *representation*, i.e., suitable data structures for genotypes, for phenotypes, for the genotype-phenotype mapping (GPM) translating the former to the latter, as well as efficient search operations [26,75,85]. Such representations are usually compact [26,74] and unbiased [8,64]. The GPMs should be surjective [64], injective [74], and consistent [64]. The search operations should be complete [80,85,88], should not destroy good solution components [74], and genes jointly encoding one phenotypic trait should be co-located in the genotype [69]. Indirect representations form a special class [6]: Developmental ontogenic mappings [18,63], for instance, improve the scalability of an EA, i.e., the ability to solve larger-scale problems, by translating small genotype data structures to much larger phenotype data structures via an injective, non-surjective, iterative GPM process. Several pitfalls when solving optimization problems can be avoided by good representation design [86,88].

Although the concept of genotypes and phenotypes stems from the EC field, it can also be applied in local search methods. The wide variety of available representations, ranging from bit strings, permutations, real vectors, to tree and graph data structures is therefore not a unique feature or advantage of EAs. The only distinct representation component of an EA is the binary crossover operator. As pointed out in the previous section, crossover may significantly improve the speed of an optimization process, but may also contribute to premature convergence.

2.6 Advanced methods

There exist several advanced techniques to improve the performance of an EA, and they often try to prevent premature convergence. Sharing, niching [27,36] and clearing [65] try to prevent the population of an EA from collapsing to identical individuals. The population can also be clustered in order to additionally explore multiple optima at once [58,87]. Instead of optimizing the objective function directly, FUSS [41], Novelty Search [50], and FFA [89] try to find only new solution characteristics, where *new* may be defined with respect to the objective function and history of the search.

Other methods aim to improve the speed of the optimization process. Linkage learning [32] and variable interaction learning [15] try to detect the *a priori* unknown relationships between the decision variables of a problem and use this information for a more targeted search. If an objective function is at least partially separable, this can be exploited by cooperative co-evolutionary approaches [15,67]. Self-adaptation methods for the population size or mutation rate also aim to improve the search speed (but may cause premature convergence [77]).

Most of the above methods require a population size significantly larger than 1 to work. They therefore represent another unique feature (besides crossover) distinguishing EAs from local search. In this paper, we focus on pure EAs and cannot consider the full spectrum of this variety of methods. Nevertheless, different ways to adapt the population size will be analyzed both in our literature study (Sect. 3.3) and experiments (Sect. 5).

2.7 Outperform in general

There is another 'hidden' problem with the initial statement: It implicitly generalizes. It does not limit the type of optimization problems to some specific features. The No Free Lunch Theorem (NFLT) for optimization [96] implies that no optimization algorithm can consistently outperform another one over all possible optimization tasks. This also implies that EAs can outperform local search algorithms only on problems with specific features.

2.8 The nature of an EA

We consider a $(\mu + \lambda)$ EA, which starts with a set of λ randomly created candidate solutions. Out of these, the best $\mu \leq \lambda$ solutions will be selected as "parents" of the second generation: λ offspring are created by applying either a unary (mutation) or binary (crossover) operator to the parents. From then on, the μ best individuals are selected from the λ offspring and their μ parents in each generation.

We already have established that such an EA should behave like random sampling for $\mu \to \infty$. A $(1 + 1)$ EA, on the other hand, *is* a greedy hill climber [91,92], i.e., the simplest local search, which puts all of its computational effort on exploitation. Both random sampling and hill climbing usually should be outperformed by more advanced local search algorithms such as Tabu Search [7,24,25] or Simulated Annealing [13,46,72] on practically relevant problems. An interesting question, which we will investigate in the remainder of this article, is whether there is an optimal population size (i.e., a setting of μ and λ in between) at which the EA can outperform local search?

3 Related work: population sizes in EAs

In the previous sections, we have provided a set of basic considerations on the nature of EAs under extreme population settings. We will now review the literature on population sizes of EAs, which we divide into three groups: studies that analyze the behavior of EAs from a mathematical perspective, studies that focus on experimental investigations, and those that consider adaptation of the population size.

3.1 Theoretical analysis

Most theoretical approaches to the relationship between the population size and algorithm performance consider relatively narrow classes of problems, such as OneMax and bit-string based representations. They usually analyze under which conditions the expected running time to discover the global optimum is polynomial, i.e., the EA is *efficient* [42], and when it becomes exponential. Yao [99] called problem instances that can be solved efficiently with an EA *EA-easy* and those that cannot *EA-hard*.

Several researchers have studied the impact of population sizes for EAs without recombination: Jansen and Wegener [43] defined an example problem that can be solved efficiently by an EA with a population size above a certain threshold, whereas a $(1 + 1)$ EA, i.e., local search, needs super-polynomial runtime in expectation. Similar to Jansen and Wegener, Witt [94,95] applied an EA with fitness proportionate selection to an example problem. He, too, found that a population size of 1 leads to exponential runtime in expectation, whereas an EA with a population size above a problem scale dependent threshold is efficient. Interestingly, he also showed that with a slight modification to the objective function, opposite results

can be obtained. That is, EAs with population sizes above a problem-dependent threshold become inefficient while $(1 + 1)$ EAs have polynomial expected running times[1].

He and Yao [33] compared $(1 + 1)$ EAs with $(\mu + \mu)$ EAs (again without recombination) and found that, in some cases, EAs with populations $(\mu > 1)$ can solve a problem efficiently, while $(1 + 1)$ EAs cannot. In cases when (1+1) EAs are efficient in solving a problem, a high selection pressure may render a population-based EA inefficient. Chen et al. [14] even showed that an EA without recombination and large populations loses the ability to solve a TRAPZEROS problem in polynomial time.

Jansen et al. [44] analyzed a $(1 + \lambda)$ EA on the ONEMAX and LEADINGONES problems and found that $\lambda > 1$ does not bring a performance benefit over $\lambda = 1$ on these problems. They further provided empirical results indicating that some harder problems may require $\lambda > 1$ to discover the optimum.

Jägersküpper and Storch [42] found that a $(1, \lambda)$ EA will outperform a $(1 + \lambda)$ EA on the bit string domain if λ is logarithmic in the problem scale n. On unimodal problems, the $(1, \lambda)$ EA with smaller populations becomes inefficient. For larger populations, both the EAs behave similarly. Rowe and Sudholt [76] derived a threshold below which the running time of a $(1, \lambda)$ EA is exponential and above which it is polynomial in expectation on the ONEMAX problem. They also concluded that for other unimodal problems, the threshold may be higher.

Storch [81,82] investigated a $(\mu+1)$ EA avoiding duplicates in its population and without recombination on four example problems, including ONEMAX. They showed that for some problems, such an EA is only efficient if μ is greater than a certain polynomial in the problem scale. Li and Wang [51] found a relationship between the optimal population size of an EA and the encoding length (i.e., the problem scale n). "Optimal" here implies that the EA performs worse for either larger or smaller populations.

Gao [23] defined bounds for the population size of an EA with fitness proportionate selection above which the actual sampling rates for schemas closely resemble those theoretically expected based on Holland's Schema theorem. These bounds are based on the assumption that fitness proportionate selection and the objective function together would nicely steer the EA towards good solutions, which is not necessarily the case (as in the counter examples in [95]).

This (surely incomplete) overview on theoretical analysis of the population size parameter complements our ideas from the previous sections. A $(1 + 1)$ EA is a hill climber, which may be able to outperform 'true' EAs on some unimodal problems. The existence of lower bounds for the required population size to render an EA efficient on other problems, together with the fact that a $(\infty + \lambda)$ EA equals random sampling, support the assumption that there should be optimal population size settings in between.

An interesting issue is that problems one would normally use an EA for in practice are often *NP*-hard, while problems in the above-mentioned studies seem to be rather easy. As a result, these studies considered only the expected runtime to find the optimum, which one would intuitively expect to be exponential for *NP*-hard problems such as the TSP[2]. When dealing with such problems, we are therefore interested in the (functional) relationship between consumed runtime and approximation quality. In order to obtain such relationships, at least approximately, experimental investigations appear to be necessary.

[1] This could be a manifestation of the NFLT.

[2] The question of whether there is a population size threshold above or below which the expected running time becomes polynomial would then, again intuitively, be answered with *probably not*.

3.2 Experimental analysis

As results from theory are still limited, we now provide an overview of experimental studies investigating the population size parameter.

Sarker and Kazi [78] applied an EA to a two-stage transportation problem and tested population sizes from 50 to 2000. They found that larger populations provide better results, but require more objective function evaluations (*FE*s). They observed that the increase in required *FE*s was sub-linear to the increase of population sizes. We argue that it is necessary to compare the solutions that different algorithm settings achieve using the *same* amount of runtime or *FE*s.

Gotshall and Rylander [28] provided an explanation for optimal population size settings: *"We show that increasing the population size increases the accuracy of the [EA]. Increasing population size also causes the number of generations to converge to increase. The optimal population for a given problem is the point of inflection where the benefit of quick convergence is offset by increasing inaccuracy."* They performed 250 runs per population size and counted how many of them contain a correct solution on three different benchmark problems, including OneMax and the *NP*-hard Maximum Clique problem. As for the termination criterion, they (seemingly) [3] used the number of generations that all individuals in the population become synonymous. We compare the improvement of approximation quality of different setups over time and thus, neither consider an explicit point of termination nor focus on finding the correct solutions only.

Another reason why optimal settings for the population size parameter could exist is given by Abu Bakar and Mahadzir [1]: *"If the population size is too large, the selective pressure will be reduced considerably and this will make the search ineffective. On the other hand if the population size is too small, selective pressure will be strong and this exploits certain individuals that will result in a premature convergence."* In their experiments on Sudoku puzzles with population sizes in the range of 10–1000, the lowest CPU time and the lowest number of generations, mutations, and crossover applications to find a correct solution were all achieved with a population of 500 individuals.

Hidalgo [35] used an EA with a fixed number of 1024 *FE*s for the OneMax and Trap functions. He tested which division of these *FE*s in different runs with different population sizes yields the best results. He obtained the best results when all *FE*s were granted to a single run with the largest tested population (32 on the OneMax and 256 on Trap problems). His comparison also showed that on some problems, smaller populations yield better results earlier but are eventually outperformed by the larger populations, which we can confirm in our experiments (at least until the optimal setting is reached).

Roeva et al. [73] tested population sizes from 5 to 200 for 200 generations on an *E. Coli* Fed-batch Cultivation Model. The best results were obtained for population size 100. The computational time always increases with increasing population sizes. In our analysis, we do not only consider end results, but the progress over runtime [90].

Lin and Chen [53] investigated $(\mu + \mu)$ and (μ, μ) EAs on Path-wise Discrete Lipschitz Quasi-Basin class benchmark problems and found that small population sizes are good in terms of the *FE*s needed to discover the optimum. We, on the other hand, compare approximation quality over runtime, as we cannot generally expect to discover the global optima in TSPs or other *NP*-hard problems.

[3] *"We suspect the increase of generations to convergence is probably due to the overall increase in probability that a mutation will take place in the population and in the increased time it takes for a greater number of chromosomes to completely converge."* [28].

Costa et al. [16] investigated an EA with population sizes of 50, 100, 512, and 1000. On several numerical benchmark problems, the population size of 512 led to best results. The population size of 1000 had performed the worst. They concluded that the limited amount of 5000 *FE*s granted to each experiment is the reason behind it. On their experiments with the Royal Road problem, the EA with population size 1000 had the best results.

Piszcz and Soule [66] investigated a set of example problems for Genetic Programming (GP) and identified optimal population sizes in terms of the computational effort needed to solve the problems. Both smaller and larger sizes would lead to additional efforts required. We again argue that only focusing on final results may not unveil the full characteristics of the population size parameter. Like Gotshall and Rylander [28], Piszcz and Soule [66] also discovered a relationship between the optimal population size and the problem scale.

Hu and Banzhaf [38] considered the problem of population sizes from the perspective of the rate of evolution R_e—the rate of adaptive genetic changes being accepted into the population of a GP process applied to a Symbolic Regression problem. They tested the population sizes 200, 2000, and 20000. The observed differences were small, but larger populations led to a better average fitness. They also have a higher R_e in the first generations, while the smallest population has a higher R_e later on. In order to be compatible with the other experiments in the same work, generations were used as time measures. It should be noted that a population of 20000 individuals can perform 100 times more *FE*s than a population of 200 individuals, while the diagrams provided were for 50 generations only.

Hu et al. [40] compared a standard EA with a fixed population size to their asynchronous parallel EA. For the standard EA, they tested four different population sizes from 136 to 685 on the OneMax problem and found that it needs less time to discover the optimum for the smallest setting. On a multi-modal problem, they got a similar observation. This could either indicate that an optimal setting exists at smaller population sizes for these problems, or that the problems are simply too easy and do not require a global optimization technique.

Besides the size of the population, its initial 'content' may also have an impact on the result quality and convergence speed. Maaranen et al. [59] showed that pseudo-random initialization is fast and easy-to-use, while more sophisticated techniques such as quasi-random or SSI processes may have advantages if the 'goodness' of the final solution is valued higher than the speed of convergence during the first generations.

In summary, several studies have indicated that there should be an optimum for the population size setting in various optimization problems. It is a general consensus that small populations may lead to premature convergence, whereas large populations may waste computational resources [45,47,73]. However, many of the presented studies

- experimented with relatively small population sizes (often below 1000),
- considered generations as the time measure (thus ignoring that generations require more computational efforts with increasing population sizes),
- focused only on final results or whether the global optimum has been obtained (whereas we consider the whole progress over runtime), or
- considered problems that are not *NP*-hard.

It must be investigated whether EAs with larger populations, if given enough runtime, will always find better solutions eventually than those with smaller populations, or whether they may perform absolutely worse on some problems and under certain conditions, as some theoretical papers have indicated [14,44,94,95]. Further experiments therefore may contribute to clarifying the situations.

3.3 Adaptation of the population size

The studies discussed in the previous section indicate that optimal settings for the population size may exist and that they likely are problem instance-dependent, i.e., not known in advance. Therefore, a very active area of research is the dynamic adaptation of population sizes in EAs [20,21,56,61]. A wide variety of different approaches to population size adaptation exist, which include

1. finding a good population size based on some initial candidate solution samples [100],
2. randomly changing the population size while the algorithm is running [16] (in the absence of any prior information about good population size settings, the EA will have good population sizes at least from time to time),
3. restarting with increased population sizes [4] (if the runs with smaller population sizes can already find a sufficiently good solution, much runtime can be saved),
4. decreasing the population size [11,30] (in order to first make use of the global search capabilities to locate the basin of attraction of the global optimum and then exploit it more quickly with smaller populations), as well as
5. modifying it based on the progress of the optimization process [20,39,57].

Besides the different conceptual ideas given above, another reason for this variety is that optimization problems may have extremely different characteristics. Also, according to the NFLT [96], a strategy working on one family of problems may not work on another family of problems.

Under the assumption that having a larger population means trading in runtime for better global search capability, adapting the population size is useful only under *limited* computational budgets, i.e., in practical scenarios. If runtime is plenty, as it is often assumed when claiming that EAs will outperform local search, the benefits of adapting the population size decrease. Then, the largest population size setting available could be used statically during the whole optimization process in order to obtain at least similar or better solutions. Even the best adaptation strategy therefore does not invalidate our arguments in Sect. 2.

4 Experimental results of static population sizes

We have selected the TSP as a test bed to experimentally investigate the impact of static population size settings on the performance of an EA. The TSP is very well studied [3,29,49,52]. It is also an often-encountered sub-problem of other combinatorial or vehicle routing tasks (e.g., see [60]). A TSP instance can be defined by n cities and the distances between them. The goal is to find the shortest round trip tour visiting each city and returning back to its starting point. This optimization version of the TSP is *NP*-hard [29].

We used the *TSP Suite* [90] to benchmark a $(\mu+\lambda)$ EA. We selected the path representation [48] for tours, where permutation $(1, 3, 2)$ in a 3-city problem defines a tour first visiting city 1, then city 3, after that city 2, and finally returning to city 1. Other representations such as satellite lists [62], -trees [70] and 2-level trees [22,70] are more efficient in terms of runtime complexity of the search operations processing them. Nevertheless, the path representation is very simple and will lead to more easily reproducible results and qualitatively similar conclusions.

We tested 19 powers of 2 for μ, ranging from 1 to 262144, and set $\lambda = 2\mu$. The mutation operator randomly chooses between four operations, as defined in [63,90]: swapping two

cities in a tour, reverting a sub-sequence of a tour, or rotating a sub-sequence one step to the left or right. We used the well-known Edge crossover [93] at a crossover rate of 1/3.

Due to memory requirements of the largest population setting, we have limited our analysis to the 68 smallest-scale *TSPLib* instances [71], ranging from *burma14* with 14 cities to *si535* with 535 cities. We compared the EA with a random walk based on swap moves, as well as a hill climber and the best pure local search algorithm from [90], multi-neighborhood search[4] (MNS), both using the same four unary operators as the EA. For each configuration and benchmark instance, we performed 30 independent runs.

4.1 Global ranking

The *TSP Suite* automates most of the experiment execution, and it also has an evaluator component for computing a range of statistics and aggregating them into a global ranking [90]. This ranking can provide a first glance of the performance relationship between different algorithms and setups. For the experiment described above we obtained:

MNS (rank 1), (2048 + 4096) EA (2), (4096 + 8192) EA (3), (1024 + 2048) EA (4), (512 + 1024) EA (4), (8192 + 16384) EA (6), (128 + 256) EA (7), (64 + 128) EA (8), (256+512) EA (9), (1+2) EA (10), (32+64) EA (11), (8+16) EA (12), hill climbing (12), (16+32) EA (14), (4+8) EA (15), (16384+32768) EA (16), (2+4) EA (17), (32768+65536) EA (18), (65536+131072) EA (19), (131072+262144) EA (20), (262144+524288) EA (21), and random walk (22).

The automated procedure concludes that the local search algorithm MNS can outperform all EA setups. The best EA performance is detected around the (2048+4096) EA. Generally, EAs tend to perform worse the more their population sizes differ from this setting. EAs with really large populations perform the worst among the EAs, but are still better than the random walk. The hill climber ranks roughly similar to EAs with smaller populations.

In the following subsections, we will explore the experimental results in more detail and verify this automatically obtained ranking. We use F, defined as the amount that a tour is longer than the optimal tour divided by that optimal tour length, as a quality measure [90]. $F = 0$ corresponds to an optimal tour and a tour that is 10 % longer than the optimal one has $F = 0.1$.

4.2 Progress over time

EAs and local search algorithms are *anytime algorithms* [10], meaning that they can provide an approximate solution for a problem at any time during their course. The approximation quality may improve if more time is given. This means that it is not sufficient to just compare their final results, as the point of termination when these results are measured would be an arbitrary choice of the researcher [90]. Thus, it is necessary to compare how the algorithms progress and improve their approximation quality over runtime.

In Fig. 1 we plot how the quality F_b of the best tour an algorithm has discovered until a given point in time improves exemplarily for six of the benchmark cases. The time is measured in terms of the number of created solutions, i.e., in *FE*s.

In the diagrams, we blend the colors for the $(\mu + \lambda)$ EAs from blue for the (1 + 2) EA to violet for the (2048 + 4096) EA and red for the (262144 + 524288) EA. Among these setups, the convergence speed decreases steadily from the smallest population sizes to the largest ones. EAs with small populations perform very similar to the hill climber (orange),

[4] In [55,97,98], we found that an ejection chain method using the stem-and-cycle structure, the Lin–Kernighan heuristic and Tabu Search can outperform MNS, respectively. However, hybrids of MNS and EAs or Ant Colony Optimization outperform similar hybrids of these algorithms.

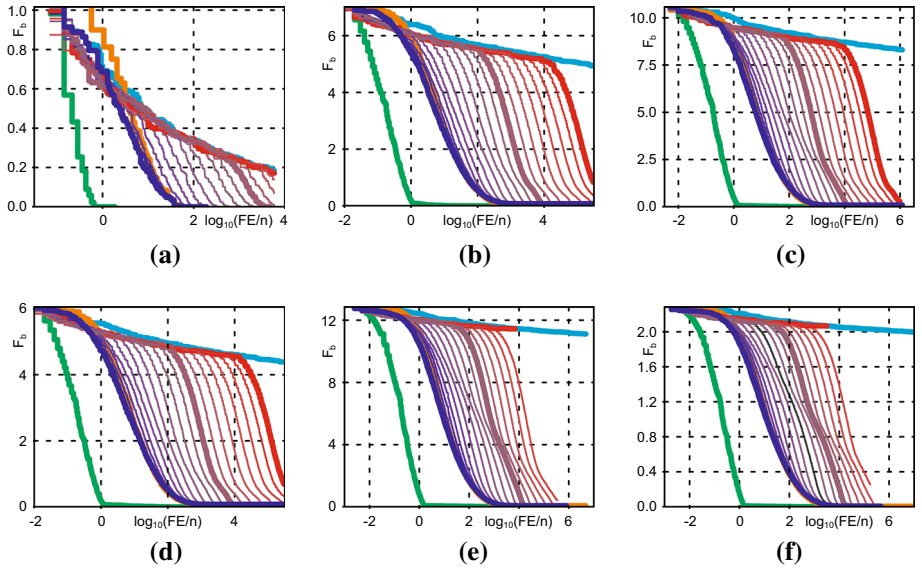

Fig. 1 The progress of algorithms in terms of the median best error F_b encountered over runtime measured in *FE*s for different benchmark instances. The hill climber performs very similar to EAs with the smallest populations, whereas the behavior of EAs with large populations, during their initial phase, is more similar to random walk. MNS performs the best. See Fig. 3a for the legend. **a** Progress over *FE* for problem *burma14* **b** Progress over *FE* for problem *KROA100* **c** Progress over *FE* for problem *KROA200* **d** Progress over *FE* for problem *rd100* **e** Progress over *FE* for problem *rd400* **f** Progress over *FE* for problem *si535*

and their curves often overlap. The hill climber is a little bit faster in terms of improving F than the $(1+2)$ EA at the beginning because of its strong exploitation. A good local search algorithm like MNS (green) has a much faster convergence rate than the EAs.

The larger the population size becomes, the longer the EA behaves like a random walk (cyan): For benchmark case *kroa100*, the curves of the $(262144 + 524288)$ EA and random walk overlap until $\log_{10}(FE/n) \approx 4$, i.e., for 1000000 *FE*s, but after that the EA performs better.

4.3 Probability to solve a problem

A question that cannot be answered from Fig. 1 is which of the setups can solve the benchmark problems or, at least, reach good approximation qualities. To reasonably well illustrate the convergence behavior, these figures often start with $F_b \geq 6$, i.e., tours seven times as long as the optimum, while acceptable deviations are at most in the low single-digit percent range.

In Fig. 2 we plot the empirical cumulative distribution function (*ECDF*) [31,37,83,90], which is defined as the fraction of runs that have reached a given goal solution quality F_t, aggregated over all benchmark cases. If the *ECDF* reaches 1, all runs have found solutions with $F_b \leq F_t$.

From Fig. 2a, we see that only about 20 % of the best setups of the $(\mu + \lambda)$ EA can find the globally optimal solutions. Figures 2b to 2d illustrate the *ECDF* in the case where we are willing to accept non-optimal solutions and F_t is increased to 0.1 over 0.01 and 0.025. The *ECDF* curves rise higher as well, reaching close to 1 for the $(2048+4096)$ EA and $F_t = 0.1$.

For all four F_t-cases, we observe some similar behavior: The *ECDF*s of the $(\mu + \lambda)$ EAs with small populations rise earlier, due to the strong exploitation ability of these setups. They

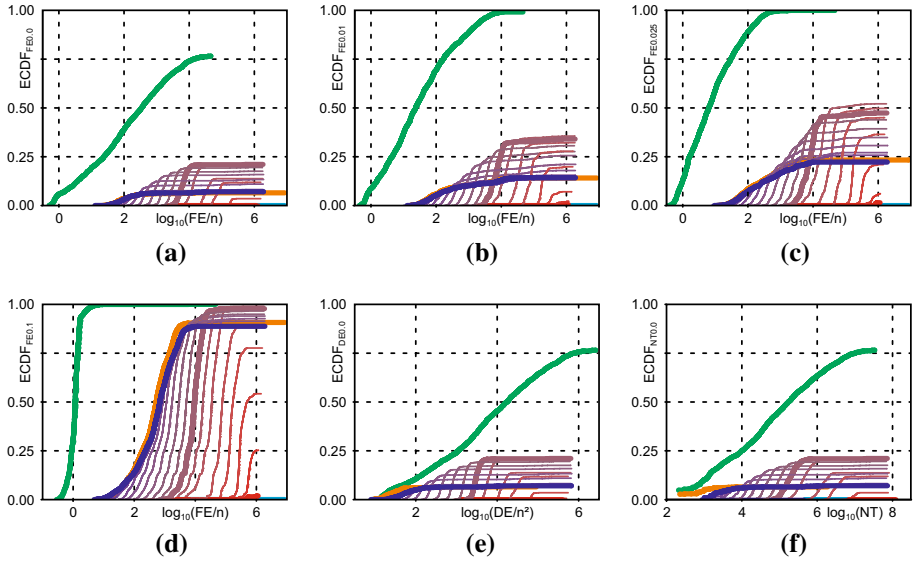

Fig. 2 The *ECDF* over different time measures (*FE*, *FE*, *NT*) and goal quality thresholds F_t. $(\mu + \lambda)$ EAs with population sizes around 2048 can solve more problems ($F_t = 0$) and achieve better solution qualities than those with small populations (which perform similar to hill climbing) or large populations (which behave similar to random walk). MNS performs the best. The algorithm behaviors are consistent over different time measures. See Fig. 3a for the legend. **a** *ECDF* for *FE* and $F_t = 0$ **b** *ECDF* for *FE* and $F_t = 0.01$ **c** *ECDF* for *FE* and $F_t = 0.025$ **d** *ECDF* for *FE* and $F_t = 0.1$ **e** *ECDF* for *DE* and $F_t = 0$ **f** *ECDF* for *NT* and $F_t = 0$

are more similar to the curves for the hill climber. However, they stagnate earlier as well, for the same reason. The highest *ECDF* values are reached by the (2048+4096) EA, but for larger populations, the *ECDF* drops again. The largest populations are about as likely to solve a problem to acceptable quality as a random walk. Due to the fact that they spend almost all runtime on exploration, they can solve almost no problems, even if a $F_t = 10\%$ error was admissible.

In [90], we outlined that measuring the runtime only in *FE*s may be unfair, as one *FE* may require different computational times for different algorithms. MNS, for instance, performs an $\mathcal{O}(n^2)$-scan of its current solution, looking for several improvements at once, while a local search move in our hill climber may require $\mathcal{O}(1)$ and the crossover operator in the EA requires $\mathcal{O}(n)$ steps for creating single new solutions. In Figs. 2e and 2f, we therefore also plot the *ECDF*s for "distance evaluations" (*DE*s), a finer-grained way to count algorithm steps than *FE*s, and the "normalized runtime" (*NT*), which is proportional to the clock time passed [90]. These plots do not look significantly different from Fig. 2a, indicating that we can here limit our considerations to *FE*s as the time measure.

4.4 Knee points and scale

Another question is at what sort of solution quality the EAs converge and what the impact of the problem scale (the number n of cities of the TSP) is. For this purpose, we plot the estimated running time (*ERT*) [31,90] (y-axis) to reach different goal qualities F_t (x-axis) in Fig. 3. Each diagram therein presents aggregated results over all benchmark instances with a scale n between two given consecutive powers of 2.

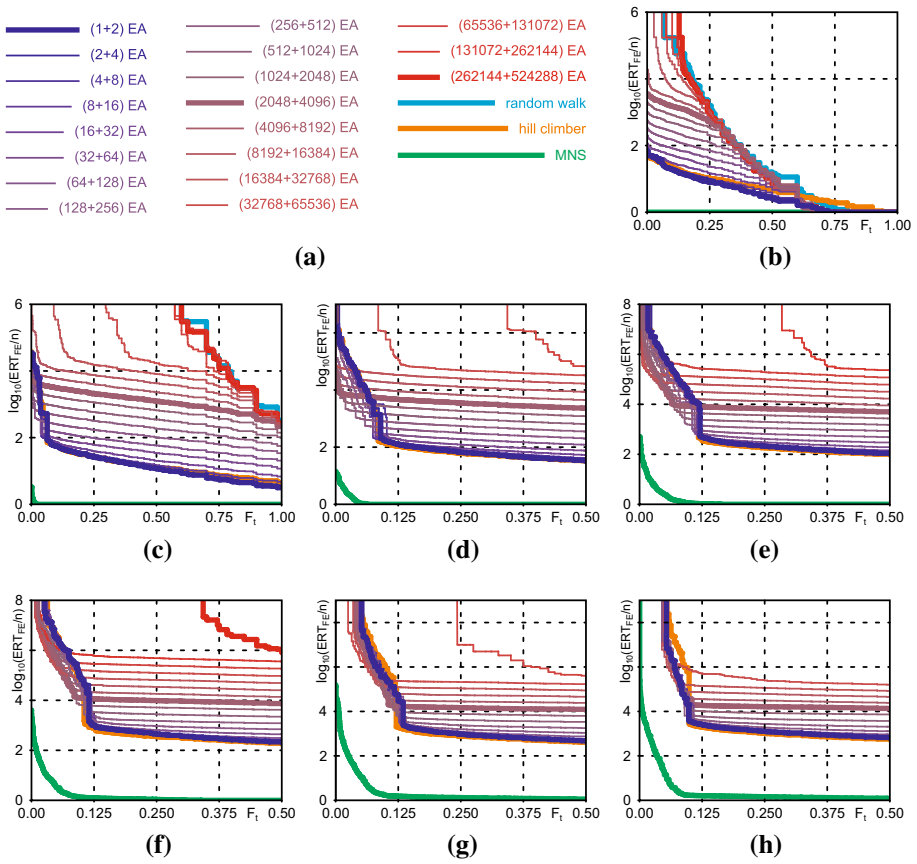

Fig. 3 The estimated running time in *FE*s (ERT_{FE}) over goal quality thresholds F_t. The *ERT* increases if the goal solution quality is improved (F_t decreases). It steeply increases at knee points. For larger-scale problems, this knee point shifts towards worse qualities (to the right). $(\mu + \lambda)$ EAs with small populations and the hill climber have similar performances and outperform $(\mu + \lambda)$ EAs with population sizes around 2048 on small-scale problems and mediocre F_t, but are slower to reach very small F_t. $(\mu + \lambda)$ EAs with large populations and the random walk cannot even achieve $F_t = 1$ (tours twice as long as the optimum) within $10^8 n$ *FE*s if n increases over 256. MNS is again the best. **a** The legend for all sub-figures in Figs. 1, 2 and 3 **b** *ERT* over 1 problem (*burma14*) with $n < 16$ **c** *ERT* over 8 problems with $16 \leq n < 32$ **d** *ERT* over 8 problems with $32 \leq n < 64$ **e** *ERT* over 17 problems with $64 \leq n < 128$ **f** *ERT* over 20 problems with $128 \leq n < 256$ **g** *ERT* over 11 problems with $256 \leq n < 512$ **h** *ERT* over 3 problems with $n \geq 512$

We find that all tested algorithms except MNS have curves characterized by 'knee points'. Solutions with tour lengths above a knee point can be obtained relatively quickly, but the estimate of the runtime needed to obtain better solutions steeply increases. For EAs with small populations and problems with $n \geq 32$, this knee point is roughly at $F_t \approx 0.125$. In comparison, for population sizes around 2048, it is located at slightly better qualities, except for the smallest and largest-scale problems. Using large populations has a very strong negative impact on the $(\mu + \lambda)$ EA performance: The knee point quickly moves towards higher F_t-values. With the increase of problem scale n (rising sub-figure index), more and more of the reddish *ERT* curves disappear from the diagrams. We can thus again confirm the similarity between the EAs with small populations and the hill climber. The visible deviations

in their *ERT* curves for $n > 256$ probably are the reason why the automated ranking by the *TSP Suite* given in Sect. 4.1 did not place the hill climber and the $(1 + 2)$ EA directly next to each other.

The problem scale n seemingly does not have much impact on the knee points, which are at solution qualities too poor for any practical relevance. It has, however, an impact on the quality threshold F_t where the *ERT* goes to infinity, i.e., below which no solutions were discovered in the experiments. This threshold moves towards worse qualities as n increases.

4.5 Final results

The *TSP Suite* terminates a run of an algorithm if it has consumed 1h of runtime, $100n^3 FEs$, $100n^4 DEs$, or discovered the global optimum, whichever happened earlier. It applies a Mann-Whitney U test with Bonferroni correction to compare the best solution qualities F_b the different setups have achieved until their termination. 'Wins' significant under an error probability of 0.02 were used to rank the algorithms and we obtained:

MNS (rank 1), $(16384 + 32768)$ EA (2), $(8192 + 16384)$ EA (3), $(32768 + 65536)$ EA (4), $(4096 + 8192)$ EA (5), $(2048 + 4096)$ EA (6), $(65536 + 131072)$ EA (7), $(1024 + 2048)$ EA (8), $(512 + 1024)$ EA (9), $(256 + 512)$ EA (10), $(128 + 256)$ EA (11), $(64 + 128)$ EA (12), $(4 + 8)$ EA (13), $(2 + 4)$ EA (14), $(1 + 2)$ EA (15), $(8 + 16)$ EA (16), $(32 + 64)$ EA (17), hill climber (18), $(16 + 32)$ EA (19), $(131072 + 262144)$ EA (20), $(262144 + 524288)$ EA (21), and random walk (22).

These results paint a different picture. The $(2048 + 4096)$ EA is taken over by four EAs with larger populations that find (statistically) significantly better results. Together with the findings from the previous sections, this means that they just need more computational efforts. The two worst-ranked EAs are those with the largest populations. However, it could be that they would rank better if more runtime was granted. This brings us back to the issue discussed in Sect. 2.1: for larger problems, the 1h of runtime limit is the critical criterion, but this is already more than what is available in most practically relevant scenarios.

This ranking indicates that the best population size of the EA is positively correlated to the total available runtime. This is supported by, e.g., Fig. 3, from which we can see that the EAs with small populations usually find solutions worse than $F_t = 0.125$ faster than the $(2048+4096)$ EA. They would be the better choice in situations where computational budgets are limited. Either way, the local search algorithm MNS is *still* better than all tested EA setups.

5 Experimental results of adaptive population sizes

On our example problem, the TSP, we found that there appears to be an optimal static population size setting above and below which the algorithm performs worse. In Sect. 3.3, we pointed out that there exist a wide variety of different concepts on how the population size of an EA may be adapted in order to improve its performance. Here we investigated seven very simple population size adaptation schemes.

5.1 Investigated setups

A good population size adaptation scheme should be able to improve the performance of the best static population size setting. Hence we carried out experiments to compare the $(2048 + 4096)$ EA and the $(512 + 1024)$ EA, both incorporated with different adaptation schemes.

The simplest adaptation scheme in our experiment randomly decides after each generation whether the population size is increased or decreased. Besides that, we tested two static adaptation schedules, which increase or decrease μ and λ after every generation, respectively. We also tested two approaches that use information about the progress of the search, one increases and the other decreases the population size only after a generation where an improved solution was found. Finally, we tested a policy that increases the population size after a generation where an improvement was found and decreases it otherwise, as well as the opposite strategy. In an adaptation step, we shrank the populations by factor $3/4$ or grew them by $4/3$, i.e., increase and decrease are inverse operations except for rounding errors.

5.2 Observations

We again used the *TSP Suite* to execute the 14 adaptive setups and compared them with the corresponding non-adaptive, static settings. Fig. 4a shows a naming scheme for this series of experiments, which is used in the following text.

The global overall ranking of the algorithms' performance provided by the *TSP Suite*, based on several different metrics, has the static $(2048+4096)$ EA (b) and $(512+1024)$ EA (n) at ranks 1 and 2, followed by the random adaption schemes nr and br at ranks 3 and 4. Then, the population size decreasing variants of the $(512+1024)$ EA are listed, thereafter followed by those of the optimal static population size. The population size increasing policies perform the worst.

In Fig. 4, we illustrate several facets that have influenced this ranking. Figure 4b shows that the static $(2048+4096)$ EA has the highest *ECDF* for $F_t = 0$ and finds the global optima of more problem instances (about 20 %) than any other setup. It is closely followed by its variant br, which changes the population size randomly. The $(512+1024)$ EA and its randomly-adapting variant can only solve about 17 % of the problems. However, they do so much faster and their *ECDF* rises earlier. The population size decreasing setups solve fewer problems. Among them, setups with name suffix d+i, marked with thicker lines in the figures, perform the best. The population size increasing setups are worst in terms of the *ECDF*.

The simple population size adaptation schemes we tested solve fewer problem instances than the corresponding static setups on the TSP example we used. The setups that randomly change the population size are likely to have, on average, the same population size as their static counterparts. Hence their behavior is quite similar to them. Decreasing the population size seems to be a better option than increasing it.

We have already confirmed that EAs with smaller populations converge faster, but are less likely to find the global optimum. This is evident again in Figs. 4c to 4e: The adaptive setups that strictly decrease the population sizes (naming suffixes d and d+, marked with thinner lines) progress much faster than the static settings in terms of their *ECDF* if the goal is to find a tour no more than 10% longer than the optimum ($F_t = 0.1$, Fig. 4c). The static settings achieve this goal more often in the end. The population size decreasing setups are also much faster in improving their best-so-far tour lengths F_b over time, see Figs. 4d and 4e. Due to the scale of the y-axis of these figures, it is not possible to see whether they converge at a worse objective value, but this can be assumed based on Fig. 4b.

5.3 Summary

Due to the limitations of this small study and the crudeness of the tested approaches, it is not possible to make general statements about population size adaptation. Any adaptation can either increase or decrease the population (or both) and do so either according to a fixed

name: (μ + λ), after each generation	name: (μ + λ), after each generation
b: (2048+4096), unchanged / static setting	br: (2048+4096), randomly either inc or dec
bi: (2048+4096), inc	bi+: (2048+4096), inc if improvement found
bi+d: (2048+4096), inc if improvement found, else dec	bd: (2048+4096), dec
bd+: (2048+4096), dec if improvement found	bd+i: (2048+4096), dec if improvement found, else inc
n: (512 +1024), unchanged / static setting	nr: (512 +1024), randomly either inc or dec
ni: (512 +1024), inc	ni+: (512 +1024), inc if improvement found
ni+d: (512 +1024), inc if improvement found, else dec	nd: (512 +1024), dec
nd+: (512 +1024), dec if improvement found	nd+i: (512 +1024), dec if improvement found, else inc

inc. . . increased, dec. . . decreased

(a)

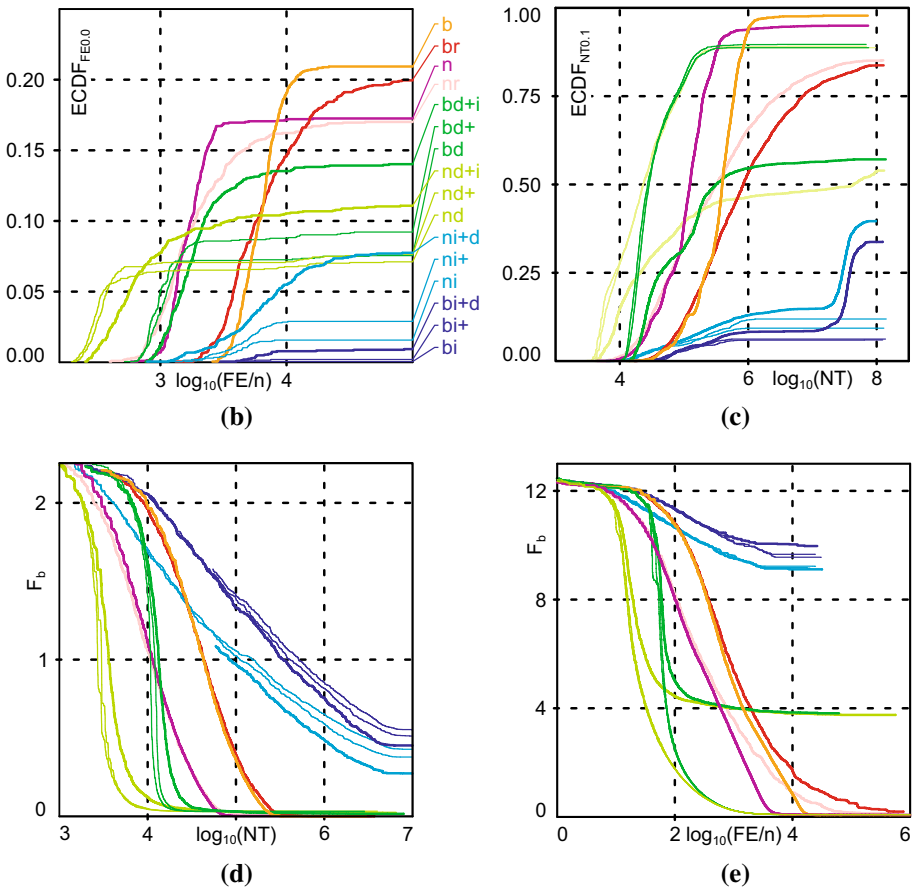

Fig. 4 *Second row* The *ECDF* over different time measures and goal quality thresholds F_t. The EAs without population size adaptation methods can solve more problems. *Third row* The progress of algorithms in terms of the median best error F_b over different runtime measures. EAs with adaptively decreasing population sizes are faster (but solve fewer problems, see Fig. 4b). **a** The legend and naming scheme for experiments with adaptive population sizes. **b** *ECDF* for *FE* and $F_t = 0$. **c** *ECDF* for *NT* and $F_t = 0.1$. **d** Progress over *NT* aggregated over 8 problems with $32 \leq n < 64$. **e** Progress over *FE* aggregated over 20 problems with $128 \leq n < 256$

schedule or based on some progress metric. Our experiment tested the simplest concepts for each of these ideas and thus provides some basic, first impression.

As pointed out in Sect. 3.3, population size adaption schemes are mainly beneficial in scenarios with limited runtime, where balancing exploration and exploitation is more crucial.

In our experiment here, however, much runtime (e.g., 1 h or $100n^3$ *FE*s) was granted in order to verify whether EAs can outperform local search if given enough runtime. It became visible that population size decreasing methods converge faster than the corresponding static setups, at the cost of worse final results. If the computational budget had been smaller, these methods would have outperformed the static settings within the budget.

Since enough runtime was granted, the static setups have achieved a better overall performance ranking. Even the liberal runtime constraints were not sufficient for the population size increasing schemes to reach better solutions as statistical tests (not detailed here) showed. Adaptive population sizing therefore may not resolve our arguments in Sect. 2 on the assumption that EAs can outperform local search. Still, they likely can lead to some tangible improvement of performance in scenarios with small computational budgets, especially if more advanced methods are used.

6 Conclusions

The two take-away messages of this work are:

1. The belief that *EAs will outperform local search if given enough runtime and using large-enough populations*, widespread in the EC community, is likely to be incorrect in many scenarios where runtime and memory are limited.
2. Instead of obtaining better results with larger populations, there likely are optimal population size settings in such scenarios above and below which the overall performance of an EA deteriorates.

Both points above can be derived with simple considerations, which we have done in Sect. 2. We also found evidence for them in the related work study as well as in our experiments.

EAs with a population size of 1 are essentially hill climbing algorithms (i.e., local search). EAs with infinitely large population sizes are random sampling algorithms. The theoretical results outlined in our literature study (Sect. 3.1) indicate that for some problems, there are lower limits of the population size only after which an EA becomes efficient in solving certain problems. For these problems, there thus should exist an optimal setting for a specific finite computational budget according to key message 2. This assumption is further substantiated by several of the experiment-oriented related studies listed in Sect. 3.2.

We also conducted a large-scale experimental study using the TSP, in which we compared the performance of EAs with population sizes spanning 19 powers of 2 with each other, with a random walk, a hill climber, and a local search algorithm. The experiments confirmed that $(\mu + \lambda)$ EAs with small populations behave similar to the hill climber, whereas those with very large populations behave similar to random walk. In between these two extremes, we found that $\mu \approx 2048$ performs the best, if $\lambda = 2\mu$ and with respect to the termination criteria in the *TSP Suite*.

In [90], we only investigated four pure EA configurations, which, with $\mu = 16$ and 128 and λ of either 64 or 256, were quite different from this setup. Nevertheless, even the (2048+4096) EA performs significantly worse than the specialized local search algorithm MNS. This is one additional piece of evidence pointing towards the correctness of our arguments regarding the claim that EAs will outperform local search if given large-enough populations and long-enough runtime.

Our literature study provides substantial support for our key messages. Our experiments alone, although extensive and conclusive, can only serve as further indicators, since

1. we only considered the TSP and gave no information about possible scenarios of other problems,
2. we only tested one representation for solutions of the TSP, the path representation, which may not be the most efficient choice,
3. we only covered a limited amount of setups, all of which used the same crossover rates and operators and $\lambda = 2\mu$,
4. we only considered $(\mu + \lambda)$ EAs[5], and
5. we did not consider advanced techniques (see Sect. 2.6), which may significantly improve the performance of an EA, or even other selection algorithms such as Tournament selection. Finally,
6. the best performing setting of an EA depends on the overall computational budget (see Sects. 4.5 and 5.3). This confirms that EAs with larger population sizes *may* provide better results if granted more runtime—which may likely be infeasible in practice.

We further analyzed some simple adaptation strategies for the population size and found that they do not lead to better performances under the given large computational budget. Some setups that adaptively decrease the population size increase the convergence speed at the cost of final result quality. They may perform significantly better than static settings if the available runtime is small. This fits to the finding that EAs with (static) smaller population sizes converge faster. It can be expected that more advanced population size adaptation schemes, based on ruggedness information [68] like [100], will improve the performance of EAs more significantly. Still, it is likely that none of them can ensure that EAs will outperform local search consistently or on the majority of practical problems.

It should be noted that our considerations focus on the combinatorial domain. Numerical optimization problems have different features, e.g., they allow for less disruptive and more targeted mutation and crossover operators. EAs may here have a salient advantage over local search methods. However, even within the combinatorial domain, there are several aspects that we want to clarify in our future work: We plan to investigate whether similar observations can be made in a variety of different problems and also investigate different solution representations.

The experiments conducted in this paper were performed in a scientific setup. The computational budget was chosen deliberately large to allow for a fair testing of our hypotheses. Additional experiments must be conducted to investigate the role of the population size parameter in real-world scenarios with real-world resource limitations.

Finally, similar to, e.g., Chen et al. [13], we confirmed in [55,90,97,98] that hybrid algorithms composed of an EA and a local search can consistently outperform pure local search. However, despite the maturity of the field, more research is necessary on whether, when, and why pure EAs can outperform specialized local search methods on combinatorial problems under practical conditions.

Acknowledgments The authors at UBRI would like to acknowledge support from the National Natural Science Foundation of China under Grants 61175065, 61329302, and 61150110488, the Fundamental Research Funds for the Central Universities, the Technological Fund of Anhui Province for Outstanding Youth under Grant 1108085J16, the Special Financial Grant 201104329 from the China Postdoctoral Science Foundation, the Chinese Academy of Sciences (CAS) Fellowship for Young International Scientists 2011Y1GB01, and the European Union 7th Framework Program under Grant 247619. The third author would like to acknowledge support from the University of Newcastle Faculty of Science and Information Technology's Strategic Initiatives

[5] We also tested (μ, λ) EAs, but like Lin and Chen [53], we found that these performed worse than the $(\mu + \lambda)$ EAs, so we excluded them from the discussion for the sake of brevity.

Research Fund (Grant 10.31415). The experiments reported in this work were executed on the supercomputing system in the Supercomputing Center of University of Science and Technology of China.

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Abu Bakar, N., Mahadzir, M.F.: The impact of population size on knowledge acquisition in genetic algorithms paradigm: finding solutions in the game of sudoku. In: Baharom, F., Mahmuddin, M., Yusof Y, Ishak, W.H.W., Saip, M.A. (eds.) Knowledge Management: Theory, Research, and Practice. Proceedings of the 5th International Conference on Knowledge Management (KMICe'10), Universiti Utara Malaysia (UUM), pp. 644–648. Sintok, Kedah, Malaysia (2010)
2. Angeline, P.J.: Subtree crossover: building block engine or macromutation? In: Koza, J.R., Deb, K., Dorigo, M., Fogel, D.B., Garzon, M.H., Iba, H., Riolo, R.L. (eds.) Proceedings of the Second Annual Conference on Genetic Programming (GP'97), pp. 9–17. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1997)
3. Applegate, D.L., Bixby, R.E., Chvátal, V., Cook, W.J.: The Traveling Salesman Problem: A Computational Study. Princeton Series in Applied Mathematics. Princeton University Press, Princeton, NJ (2007)
4. Auger, A., Hansen, N.: A restart cma evolution strategy with increasing population size. In: Corne, D.W., Michalewicz, Z., McKay, R.I., Eiben ÁE, Fogel, D.B., Fonseca, C.M., Raidl, G.R., Tan, K.C., Zalzala, A.M.S. (eds.) Proceedings of the IEEE Congress on Evolutionary Computation (CEC'05), pp. 1769–1776. IEEE Computer Society, Piscataway, NJ, USA (2005). doi:10.1109/CEC.2005.1554902
5. Bäck, T., Fogel, D.B., Michalewicz, Z. (eds.): Handbook of Evolutionary Computation. Computational Intelligence Library. Oxford University Press Inc, New York (1997)
6. Bentley, P.J., Kumar, S.P.: The ways to grow designs: a comparison of embryogenies for an evolutionary design problem. In: Banzhaf, W., Daida, J.M., Eiben, Á.E., Garzon, M.H., Honavar, V., Jakiela, M.J., Smith, R.E. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99), pp. 35–43. Morgan Kaufmann Publishers Inc., San Francisco (1999)
7. Berend, D., Korach, E., Zucker, S.: Tabu search for the BWC problem. J. Glob. Optim. **54**(4), 649–667 (2012). doi:10.1007/s10898-011-9783-1
8. Beyer, H., Schwefel, H.: Evolution strategies—a comprehensive introduction. Nat Comput Int J **1**(1), 3–52 (2002). doi:10.1023/A:1015059928466
9. Birbil, Şİ., Fang, S., Sheu, R.: On the convergence of a population-based global optimization algorithm. J. Glob. Optim. **30**(2–3), 301–318 (2004). doi:10.1007/s10898-004-8270-3
10. Boddy, M.S., Dean, T.L.: Solving time-dependent planning problems. Tech. Rep. CS-89-03, Brown University, Department of Computer Science, Providence, RI, USA (1989)
11. Brest, J., Maučec, M.S.: Self-adaptive differential evolution algorithm using population size reduction and three strategies. Spacecom **15**(11), 2157–2174 (2011). doi:10.1007/s00500-010-0644-5
12. Campana, E.F., Fasano, G., Pinto, A.: Dynamic analysis for the selection of parameters and initial population, in particle swarm optimization. J. Glob. Optim. **48**(3), 347–397 (2010). doi:10.1007/s10898-009-9493-0
13. Chen, D., Lee, C., Park, C., Mendes, P.: Parallelizing simulated annealing algorithms based on high-performance computer. J. Glob. Optim. **39**(2), 261–289 (2007). doi:10.1007/s10898-007-9138-0
14. Chen, T., Tang, K., Chen, G., Yao, X.: A large population size can be unhelpful in evolutionary algorithms. Theor. Comput. Sci. **436**, 54–70 (2012). doi:10.1016/j.tcs.2011.02.016
15. Chen, W., Weise, T., Yang, Z., Tang, K.: Large-scale global optimization using cooperative coevolution with variable interaction learning. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) Proceedings of the 11th International Conference on Parallel Problem Solving From Nature, Part 2 (PPSN'10-2), Springer-Verlag GmbH, Berlin, Germany. Lecture Notes in Computer Science (LNCS), vol. 6239, pp. 300–309 (2010). doi:10.1007/978-3-642-15871-1_31
16. Costa, J.C., Tavares, R., Da Rosa, A.C.: An experimental study on dynamic random variation of population size. In: IEEE International Conference on Systems, Man, and Cybernetics - Human Communication and Cybernetics (SMC'99), IEEE Computer Society, Piscataway, NJ, USA (1999). doi:10.1109/ICSMC.1999.814161
17. De Jong, K.A.: Evolutionary Computation: A Unified Approach, Bradford Books, vol. 4. MIT Press, Cambridge (2006)

18. Devert, A., Weise, T., Tang, K.: A study on scalable representations for evolutionary optimization of ground structures. Evol. Comput. **20**(3), 453–472 (2012). doi:10.1162/EVCO_a_00054
19. Doerr, B., Happ, E., Klein, C.: Crossover can provably be useful in evolutionary computation. Theor. Comput. Sci. (2010). doi:10.1016/j.tcs.2010.10.035
20. Eiben ÁE, Marchiori, E., Valkó, V.A.: Evolutionary algorithms with on-the-fly population size adjustment. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria JA, Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H. (eds.) Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII), Springer-Verlag GmbH, Berlin, Germany. Lecture Notes in Computer Science (LNCS), vol. 3242/2004, pp. 41–50 (2008). doi:10.1007/978-3-540-30217-9_5
21. Fernandes, C., Da Rosa, A.C.: Self-regulated population size in evolutionary algorithms. In: Runarsson, T.P., Beyer, H., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) Proceedings of 9th International Conference on Parallel Problem Solving from Nature (PPSN IX), Springer-Verlag GmbH, Berlin, Germany. Lecture Notes in Computer Science (LNCS), vol. 4193/2006, pp. 920–929 (2006). doi:10.1007/11844297_93
22. Fredman, M., Johnson, D., McGeoch, L., Ostheimer, G.: Data structures for traveling salesman. J. Algorithms **18**, 432–479 (1995). doi:10.1006/jagm.1995.1018
23. Gao, Y.: Population size and sampling complexity in genetic algorithms. Tech. rep., University of Alberta, Department of Computer Science, Edmonton, Alberta, Canada (2003)
24. Glover, F.W.: Tabu search—part I. ORSA J. Comput. **1**(3), 190–206 (1989). doi:10.1287/ijoc.1.3.190
25. Glover, F.W.: Tabu search—part II. ORSA J. Comput. **2**(1), 190–206 (1990). doi:10.1287/ijoc.2.1.4
26. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Longman Publishing Co. Inc, Boston (1989). doi:10.1007/3-540-32444-5
27. Goldberg, D.E., Richardson, J.T.: Genetic algorithms with sharing for multimodal function optimization. In: Grefenstette, J.J. (ed.) Proceedings of the Second International Conference on Genetic Algorithms and their Applications (ICGA'87), pp. 41–49. Lawrence Erlbaum Associates Inc, Mahwah, NJ, USA (1987)
28. Gotshall, S., Rylander, B.: Optimal population size and the genetic algorithm. In: Proceedings of the 2002 WSEAS International Conferences: 2nd IMCCAS; 2nd ISA; 2nd SOSM; 4th MEM, World Scientific and Engineering Academy and Society (WSEAS), pp. 2151–2155. Greece, Athens (2002)
29. Gutin, G.Z., Punnen, A.P. (eds.): The Traveling Salesman Problem and its Variations, Combinatorial Optimization, vol. 12. Kluwer Academic Publishers, Norwell (2002). doi:10.1007/b101971
30. Hallam, J.W., Akman, O., Akman, F.: Genetic algorithms with shrinking population size. Comput. Stat. **25**(4), 691–705 (2010). doi:10.1007/s00180-010-0197-1
31. Hansen, N., Auger, A., Finck, S., Ros, R.: Real-parameter black-box optimization benchmarking: experimental setup. Tech. rep., Université Paris Sud, Institut National de Recherche en Informatique et en Automatique (INRIA) Futurs, Équipe TAO, Orsay, France (2012)
32. Harik, G.R.: Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms. PhD thesis, University of Michigan, Ann Arbor, MI, USA (1997)
33. He, J., Yao, X.: From an individual to a population: an analysis of the first hitting time of population-based evolutionary algorithms. IEEE Trans. Evol. Comput. **6**(5), 495–511 (2002). doi:10.1109/TEVC.2002.800886
34. Helsgaun, K.: General k-opt submoves for the Lin–Kernighan TSP heuristic. Math. Program. Comput. **1**(2–3), 119–163 (2009). doi:10.1007/s12532-009-0004-6
35. Hidalgo, J.I.: Balancing the computation effort in genetic algorithms. In: Corne, D.W., Michalewicz Z, McKay, R.I., Eiben ÁE, Fogel, D.B., Fonseca, C.M., Raidl, G.R., Tan, K.C., Zalzala AMS (eds.) Proceedings of the IEEE Congress on Evolutionary Computation (CEC'05), IEEE Computer Society, Piscataway, NJ, USA, pp. 1645–1652 (2005). doi:10.1109/CEC.2005.1554886
36. Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. University of Michigan Press, Ann Arbor (1975)
37. Hoos, H.H., Stützle, T.: Evaluating las vegas algorithms: pitfalls and remedies. In: Cooper, G.F., Moral, S. (eds.) Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI'98), Morgan Kaufmann Publishers Inc., San Francisco, C.A., USA, pp. 238–245, also published as Technical Report "Forschungsbericht AIDA-98-02" of the Fachgebiet Intellektik, Fachbereich Informatik, Technische Hochschule Darmstadt, Germany (1998)
38. Hu, T., Banzhaf, W.: Nonsynonymous to synonymous substitution ratio k_a/k_s: measurement for rate of evolution in evolutionary computation. In: Proceedings of 10th International Conference on Parallel Problem Solving from Nature (PPSN X), Springer-Verlag GmbH, Berlin, Germany. Lecture Notes in Computer Science (LNCS), vol. 5199/2008, pp. 448–457 (2008). doi:10.1007/978-3-540-87700-4_45
39. Hu, T., Banzhaf, W.: The role of population size in rate of evolution in genetic programming. In: Vanneschi, L., Gustafson, S.M., Moraglio, A., de Falco, I., Ebner, M. (eds.) Proceedings of the 12th

European Conference on Genetic Programming (EuroGP'09), Springer-Verlag GmbH, Berlin, Germany. Lecture Notes in Computer Science (LNCS), vol. 5481/2009, pp. 85–96 (2009)
40. Hu, T., Harding, S., Banzhaf, W.: Variable population size and evolution acceleration: a case study with a parallel evolutionary algorithm. Genet. Program. Evol. Mach. **11**(2), 205–225 (2010). doi:10.1007/s10710-010-9105-2
41. Hutter, M., Legg, S.: Fitness uniform optimization. IEEE Trans. Evol. Comput. **10**(5), 568–589 (2006). doi:10.1109/TEVC.2005.863127
42. Jägersküpper, J., Storch, T.: When the plus strategy outperforms the comma strategy and when not. In: Mendel, J.M., Omari, T., Yao, X. (eds.) The First IEEE Symposium on Foundations of Computational Intelligence (FOCI'07), IEEE Computer Society, Piscataway, NJ, USA, pp. 25–32 (2007). doi:10.1109/FOCI.2007.372143
43. Jansen, T., Wegener, I.: On the utility of populations in evolutionary algorithms. In: Spector, L., Goodman, E.D., Wu, A.S., Langdon, W.B., Voigt, H., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M.H., Burke, E.K. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'01), pp. 1034–1041. Morgan Kaufmann Publishers Inc, San Francisco, CA, USA (2001)
44. Jansen, T., De Jong, K.A., Wegener, I.: On the choice of the offspring population size in evolutionary algorithms. Evol. Comput. **13**(4), 413–440 (2005). doi:10.1162/106365605774666921
45. Khor, E.F., Tan, K.C., Wang, M.L., Lee, T.H.: Evolutionary algorithm with dynamic population size for multi-objective optimization. In: Proceedings of the 26th Annual Confjerence of the IEEE Industrial Electronics Society (IECON'00), IEEE, IEEE Singapore Section, Singapore, pp. 2768–2773 (2000). doi:10.1109/IECON.2000.972436
46. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by simulated annealing. Sci. Mag. **220**(4598), 671–680 (1983). doi:10.1126/science.220.4598.671
47. Koumousis, V.K., Katsaras, C.P.: A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. IEEE Trans. Evol. Comput. **10**(1), 19–28 (2006). doi:10.1109/TEVC.2005.860765
48. Larrañaga, P., Kuijpers, C.M.H., Murga, R.H., Inza, I., Dizdarevic, S.: Genetic algorithms for the travelling salesman problem: a review of representations and operators. J. Artif. Intell. Res. **13**(2), 129–170 (1999). doi:10.1023/A:1006529012972
49. Lawler, E.L.G., Lenstra, J.K., Kan, A.H.G.R., Shmoys, D.B.: The traveling salesman problem: a guided tour of combinatorial optimization. Estimation, Simulation, and Control—Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley Interscience, Chichester (1985)
50. Lehman, J., Stanley, K.O.: Abandoning objectives: evolution through the search for novelty alone. Evol. Comput. **19**(2), 189–223 (2011). doi:10.1162/EVCO_a_00025
51. Li, D., Wang, L.: A study on the optimal population size of genetic algorithm. In: Proceedings of Fourth World Congress on Intelligent Control and Automation, vol. 4 (WCICA'02), IEEE, IEEE Singapore Section, Singapore, pp. 3019–3021 (2002). doi:10.1109/WCICA.2002.1020082 (**in Chinese**)
52. Li, W.: Seeking global edges for traveling salesman problem in multi-start search. J. Glob. Optim. **51**(3), 515–540 (2011). doi:10.1007/s10898-010-9643-4
53. Lin, J., Chen, Y.: On the effect of population size and selection mechanism from the viewpoint of collaboration between exploration and exploitation. In: Proceedings of the 2013 IEEE Workshop on Memetic Computing (MC), 2013 IEEE Symposium Series on Computational Intelligence (SSCI'13), IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 16–23 (2013). doi:10.1109/MC.2013.6608202
54. Lin, S., Kernighan, B.W.: An effective heuristic algorithm for the traveling-salesman problem. Oper. Res. **21**(2), 498–516 (1973). doi:10.1287/opre.21.2.498
55. Liu, W., Weise, T., Wu, Y., Chiong, R.: Hybrid ejection chain methods for the traveling salesman problem. In: Gong, M., Pan, L., Song, T., Tang K, Zhang, X. (eds.) The 10th International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA'15), Springer-Verlag GmbH, Berlin, Germany, Communications in Computer and Information Science, vol. 562, pp. 268–282 (2015)
56. Lobo, F.G.: A review of adaptive population sizing schemes in genetic algorithms. In: Beyer, H., O'Reilly, U., Arnold, D.V., Banzhaf, W., Blum, C., Bonabeau, E.W., Cantú-Paz, E., Dasgupta, D., Deb, K., Foster JA, de Jong, E.D., Lipson, H., Llorà, X., Mancoridis, S., Pelikan, M., Raidl, G.R., Soule, T., Watson, J., Zitzler, E. (eds.) Proceedings of Genetic and Evolutionary Computation Conference (GECCO'05), ACM Press, New York, NY, USA, pp. 228–234 (2005). doi:10.1145/1102256.1102310
57. Lu, H., Yen, G.G.: Dynamic population size in multiobjective evolutionary algorithms. In: Fogel, D.B., El-Sharkawi, M.A., Yao, X., Iba H, Marrow, P., Shackleton, M. (eds.) Proceedings of the IEEE Congress on Evolutionary Computation (CEC'02), 2002 IEEE World Congress on Computation Intelligence (WCCI'02), IEEE Computer Society Press, Los Alamitos, CA, USA, vol. 1–2, pp. 1648–1653 (2002). doi:10.1109/CEC.2002.1004489

58. Lu, Q., Yao, X.: Clustering and learning gaussian distribution for continuous optimization. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. **35**(2), 195–204 (2005). doi:10.1109/TSMCC.2004.841914
59. Maaranen, H., Miettinen, K., Penttinen, A.: On initial populations of a genetic algorithm for continuous optimization problems. J. Glob. Optim. **37**(3), 405–436 (2007). doi:10.1007/s10898-006-9056-6
60. Marinakis, Y., Migdalas, A., Pardalos, P.M.: A new bilevel formulation for the vehicle routing problem and a solution method using a genetic algorithm. J. Glob. Optim. **38**(4), 555–580 (2007). doi:10.1007/s10898-006-9094-0
61. Minetti, G.F., Alfonso, H.A.: Variable size population in parallel evolutionary algorithms. In: Kwaśnicka, H., Paprzycki, M. (eds.) Proceedings of the 5th International Conference on Intelligent Systems Design and Applications (ISDA'05), IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 350–355 (2005). doi:10.1109/ISDA.2005.99
62. Osterman, C., Rego, C.: The satellite list and new data structures for traveling salesman problems. Working Paper Series HCES-06-03, University of Mississippi, School of Business Administration, Hearin Center for Enterprise Science, University, MS, USA, (2006). http://www.akira.ruc.dk/keld/teaching/algoritmedesign_f08/Artikler/02/Osterman03
63. Ouyang, J., Weise, T., Devert, A., Chiong, R.: Sdgp: A developmental approach for traveling salesman problems. In: Proceedings of the 2013 IEEE Symposium on Computational Intelligence in Production and Logistics Systems (CIPLS'13), IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 78–85 (2013). doi:10.1109/CIPLS.2013.6595203
64. Palmer, C.C., Kershenbaum, A.: Representing trees in genetic algorithms. In: Michalewicz, Z., Schaffer, J.D., Schwefel, H., Fogel, D.B., Kitano, H. (eds.) Proceedings of the First IEEE Conference on Evolutionary Computation (CEC'94), IEEE Computer Society, Piscataway, NJ, USA, vol. 1, pp. 379–384 (1994)
65. Pétrowski, A.: A clearing procedure as a niching method for genetic algorithms. In: Jidō, K., Gakkai, S. (eds.) Proceedings of IEEE International Conference on Evolutionary Computation (CEC'96), IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 798–803 (1996). doi:10.1109/ICEC.1996.542703
66. Piszcz, A.T., Soule, T.: Genetic programming: optimal population sizes for varying complexity problems. In: Keijzer, M., Cattolico, M. (eds.) Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO'06), ACM Press, New York, NY, USA, pp. 953–954 (2006). doi:10.1145/1143997.1144166
67. Potter, M.A., De Jong, K.A.: Cooperative coevolution: an architecture for evolving coadapted subcomponents. Evol. Comput. **8**(1), 1–29 (2000). doi:10.1162/106365600568086
68. Poursoltan, S., Neumann, F.: Ruggedness quantifying for constrained continuous fitness landscapes. In: Datta, R., Deb, K. (eds.) Evolutionary Constrained Optimization, Infosys Science Foundation Series, chap 2. Springer, India, pp. 29–50 (2015). doi:10.1007/978-81-322-2184-5_2
69. Radcliffe, N.J.: The algebra of genetic algorithms. Ann. Math. Artif. Intell. **10**(4), 339–384 (1994). doi:10.1007/BF01531276
70. Rego, C., Gamboa, D., Glover, F., Osterman, C.: Traveling salesman problem heuristics: leading methods, implementations and latest advances. Eur. J. Oper. Res. **2011**, 427–441 (2011). doi:10.1016/j.ejor.2010.09.010
71. Reinelt, G.: TSPLIB—a traveling salesman problem library. ORSA J. Comput. **3**(4), 376–384 (1991). doi:10.1287/ijoc.3.4.376
72. Robini, M.C., Reissman, P.: From simulated annealing to stochastic continuation: a new trend in combinatorial optimization. J. Glob. Optim. **56**(1), 185–215 (2013). doi:10.1007/s10898-012-9860-0
73. Roeva, O., Fidanova, S., Paprzycki, M.: Influence of the population size on the genetic algorithm performance in case of cultivation process modelling. In: Ganzha, M., Maciaszek, L., Paprzycki, M. (eds.) Proceedings of the 2013 Federated Conference on Computer Science and Information Systems (FedCSIS'13), pp. 371–376. IEEE, IEEE Singapore Section, Singapore (2013)
74. Ronald, S.: Robust encodings in genetic algorithms: a survey of encoding issues. In: Bäck, T., Michalewicz, Z., Yao, X. (eds.) IEEE International Conference on Evolutionary Computation (CEC'97), IEEE Computer Society, Piscataway, NJ, USA, pp. 43–48 (1997). doi:10.1109/ICEC.1997.592265
75. Rothlauf, F.: Representations for Genetic and Evolutionary Algorithms, Studies in Fuzziness and Soft Computing, vol. 104, 2nd edn. Springer, Heidelberg (2006)
76. Rowe, J.E., Sudholt, D.: The choice of the offspring population size in the (1,λ) ea. In: Soule, T., Moore, J.H. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'12), ACM, New York, NY, USA, pp. 1349–1356 (2012). doi:10.1145/2330163.2330350
77. Rudolph, G.: Self-adaptive mutations may lead to premature convergence. IEEE Trans. Evol. Comput. **5**(4), 410–414 (2001). doi:10.1109/4235.942534
78. Sarker, R.A., Kazi, M.F.A.: Population size, search space and quality of solution: an experimental study. In: Sarker, R.A., Reynolds, R.G., Abbass Amein, H.A., Tan, K.C., McKay, R.I., Essam, D.L., Gedeon,

T. (eds.) Proceedings of the IEEE Congress on Evolutionary Computation (CEC'03), IEEE Computer Society, Piscataway, NJ, USA, pp. 2011–2018 (2003). doi:10.1109/CEC.2003.1299920
79. Schutte, J.F., Groenwold, A.A.: A study of global optimization using particle swarms. J. Glob. Optim. **31**(1), 93–108 (2005). doi:10.1007/s10898-003-6454-x
80. Skubch, H.: Hierarchical Strategy Learning for FLUX Agents: An Applied Technique. VDM Verlag Dr. Müller AG und Co, KG, Saarbrücken (2006)
81. Storch, T.: On the choice of the population size. In: Deb, K., Poli, R., Banzhaf, W., Beyer, H., Burke, E.K., Darwen, P.J., Dasgupta, D., Floreano, D., Foster, J.A., Harman, M., Holland, O.E., Lanzi, P.L., Spector, L., Tettamanzi, A.G.B., Thierens, D. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference, Part I (GECCO'04), Springer-Verlag GmbH, Berlin, Germany. Lecture Notes in Computer Science (LNCS), vol. 3102/2004, pp. 748–760 (2004)
82. Storch, T.: On the choice of the parent population size. Evol. Comput. **16**(4), 557–578 (2008). doi:10.1162/evco.2008.16.4.557
83. Tompkins, D.A.D., Hoos, H.H.: Ubcsat: An implementation and experimentation environment for SLS algorithms for SAT and MAX-SAT. In: Hoos, H.H., Mitchell, D.G. (eds.) Revised Selected Papers from the Seventh International Conference on Theory and Applications of Satisfiability Testing (SAT'04), Springer-Verlag GmbH, Berlin, Germany, Lecture Notes in Computer Science (LNCS), vol. 3542, pp. 306–320 (2004). doi:10.1007/11527695_24
84. Vincenti, A., Ahmadian, M.R., Vannucci, P.: BIANCA: a genetic algorithm to solve hard combinatorial optimisation problems in engineering. J. Glob. Optim. **48**(3), 399–421 (2010). doi:10.1007/s10898-009-9503-2
85. Weise, T.: Global Optimization Algorithms—Theory and Application. it-weise.de (self-published), Germany (2009)
86. Weise, T., Zapf, M., Chiong, R., Nebro Urbaneja, A.J.: Why is optimization difficult? In: Chiong, R. (ed.) Nature-Inspired Algorithms for Optimisation, Studies in Computational Intelligence, vol. 193, chap 1. Springer-Verlag, Berlin/Heidelberg, pp. 1–50 (2009). doi:10.1007/978-3-642-00267-0_1
87. Weise, T., Niemczyk, S., Chiong, R., Wan, M.: A framework for multi-model edas with model recombination. In: Applications of Evolutionary Computation. Proceedings of EvoApplications 2011: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM, and EvoSTOC, Part 1 (EvoAPPLICATIONS'11), Springer-Verlag GmbH, Berlin, Germany. Lecture Notes in Computer Science (LNCS), vol. 6624, pp. 304–313 (2011). doi:10.1007/978-3-642-20525-5_31
88. Weise, T., Chiong, R., Tang, K.: Evolutionary optimization: pitfalls and booby traps. J. Comput. Sci. Technol. **27**(5), 907–936 (2012). doi:10.1007/s11390-012-1274-4
89. Weise, T., Wan, M., Tang, K., Wang, P., Devert, A., Yao, X.: Frequency fitness assignment. IEEE Trans. Evol. Comput. **18**, 226–243 (2013). doi:10.1109/TEVC.2013.2251885
90. Weise, T., Chiong, R., Tang, K., Lässig, J., Tsutsui, S., Chen, W., Michalewicz, Z., Yao, X.: Benchmarking optimization algorithms: an open source framework for the traveling salesman problem. IEEE Comput. Intell. Mag. **9**(3):40–52 (2014). doi:10.1109/MCI.2014.2326101, featured article and selected paper at the website of the IEEE Computational Intelligence Society (http://cis.ieee.org/)
91. Whitley, L.D.: A genetic algorithm tutorial. Tech. Rep. CS-93-103, Colorado State University, Computer Science Department, Fort Collins, CO, USA (1993)
92. Whitley, L.D.: A genetic algorithm tutorial. Stat. Comput. **4**(2), 65–85 (1994). doi:10.1007/BF00175354
93. Whitley, L.D., Starkweather, T., Fuquay, D.: Scheduling problems and traveling salesman: the genetic edge recombination operator. In: Schaffer, J.D. (ed.) Proceedings of the Third International Conference on Genetic Algorithms (ICGA'89), pp. 133–140. Morgan Kaufmann Publishers Inc, San Francisco, CA, USA (1989)
94. Witt, C.: Population size vs. runtime of a simple evolutionary algorithm. In: Reihe Computational Intelligence Collaborative Research Center 531: Design and Management of Complex Technical Processes and Systems by means of Computational Intelligence Methods CI-156/03, University of Dortmund, Dept. of Computer Science/XI, Secretary of the SFB 531, Dortmund, North Rhine-Westphalia, Germany (2003)
95. Witt, C.: Population size versus runtime of a simple evolutionary algorithm. Theor. Comput. Sci. **403**(1), 104–120 (2008). doi:10.1016/j.tcs.2008.05.011
96. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1997). doi:10.1109/4235.585893
97. Wu, Y., Weise, T., Chiong, R.: Local search for the traveling salesman problem: a comparative study. In: Ge, N., Lu, J., Wang, Y., Howard, N., Chen, P., Tao, X., Zhang, B., Zadeh, L.A. (eds.) 14th IEEE International Conference on Cognitive Informatics and Cognitive Computing (ICCI*CC'15), pp. 213–220. IEEE, Piscataway, NJ, USA (2015)
98. Xu, D., Weise, T., Wu, Y., Lässig, J., Chiong, R.: An investigation of hybrid tabu search for the traveling salesman problem. In: Gong M, Pan, L., Song, T., Tang, K., Zhang, X. (eds.) The 10th International Con-

ference on Bio-Inspired Computing: Theories and Applications (BIC-TA'15), Springer-Verlag GmbH, Berlin, Germany, Communications in Computer and Information Science, vol. 562, pp. 523–537 (2015)
99. Yao, X.: Unpacking and understanding evolutionary algorithms. In: Liu, J., Alippi, C., Bouchon-Meunier, B. (eds.) Advances in Computational Intelligence: Plenary/Invited Lectures at the IEEE World Congress on Computational Intelligence (WCCI'12), Springer-Verlag GmbH, Berlin, Germany. Lecture Notes in Computer Science (LNCS), pp. 60–76 (2012). doi:10.1007/978-3-642-30687-7_4
100. Zhang, J., Yuan, X., Buckles, B.P.: Multimodal function optimization using local ruggedness information. In: Barr, V., Markov, Z. (eds.) Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference (FLAIRS 17), pp. 380–386. AAAI Press, Menlo Park, CA, USA (2004)