Fitting Multi-Planet Transit Models to Photometric Time-Data Series by Evolution Strategies

Andreas M. Chwatal
Algorithms and Datastructures Group
Vienna University of Technology
Favoritenstraße 9-11
1040 Vienna, Austria
chwatal@ads.tuwien.ac.at

Günther R. Raidl
Algorithms and Datastructures Group
Vienna University of Technology
Favoritenstraße 9-11
1040 Vienna, Austria
raidl@ads.tuwien.ac.at

Michael Zöch
Algorithms and Datastructures Group
Vienna University of Technology
Favoritenstraße 9-11
1040 Vienna, Austria
zoech@ads.tuwien.ac.at

ABSTRACT

In this paper we present the application of an evolution strategy to the problem of detecting multi-planet transit events in photometric time-data series. Planetary transits occur when a planet regularly eclipses its host star, reducing stellar luminosity. The transit method is amongst the most successful detection methods for exoplanets and is presently performed by space telescope missions.

The goal of the presented algorithm is to find high quality fits of multi-planet transit models to observational data, which is a challenging computational task. In particular we present a method for an effective objective function evaluation and show how the algorithm can be implemented on graphics processing units. Results on artificial test data with three artificial planets are reported.

Categories and Subject Descriptors

I.2.8 [**Computing Methodologies**]: Artificial Intelligence—*Problem Solving, Control Methods, and Search*
; G.4 [**Mathematics of Computing**]: Mathematical Software—*Algorithm design and analysis*
; G.4 [**Mathematics of Computing**]: Mathematical Software—*Parallel and vector implementations*

General Terms

Algorithms, Theory

Keywords

Evolution Strategies, Exoplanets, Transits, Model Fitting, Multi-Planet Systems, GPU

1. INTRODUCTION

Due to the discovery of more than 400 planets around other stars than our sun the field of exoplanet research attained vast importance in the last decades. See [14] for a comprehensive summary. Whereas most of these planets have been detected by ground-based radial velocity measurements, the *transit-method* recently became more and more important due to space-based missions like CoRoT[1] [4, 5] and Kepler [17]. In the course of these missions efficient transit detection algorithms have been developed, mostly tailored to the detection of single-planet transit events. Beyond that, also approaches to detect multi-planet transiting systems have gained attention recently, as for instance transit-timing variation analysis due to possibly gravitational interactions of the involved planets. However, so far no system of multiple planets has been detected within photometric data sets which may have its cause in the inability of current detection algorithms to deal with such scenarios. Therefore, the development of alternative algorithmic approaches currently is an attractive and promising field of research.

This work is based on [11], where the approach of simultaneously fitting multiple transit-models by metaheuristic algorithms has been outlined first. As the performance of detection algorithms is crucial due to large amounts of data to be analyzed, we focus on a more efficient evaluation of the objective function for the therein presented approaches in this work. In addition we pursue an approach of reducing the parameter space in the case of more than two planets, and show how the algorithm can be implemented on a graphics processing unit (GPU). We call the algorithmic framework presented in this work the ESTEX algorithm (Evolutionary Search for Transiting EXoplanets).

After an informal problem description in Section 2 and giving references to the most important previous works in Section 3, our approach is presented in Sections 4 and 5. Computational results are discussed in Section 6, and issues regarding the application of the algorithm to real photometric data instances are addressed in Section 7. Conclusions are finally drawn in Section 8.

2. PROBLEM DESCRIPTION

A transiting planet periodically shadows some of the light from its host star for a short time when it moves into our line of sight to the star. During the transit the luminosity of the star is marginally reduced. By neglecting the in- and egress phases, the transit-lightcurve can be well approximated by a periodic rectangular signal. The corresponding parameters are the period p the transit occurs with, a phase offset τ, the length l of the transit, and finally the transit depth d. The latter parameter corresponds to the percentage of light from the star being shadowed by the transiting planet.

Assuming M planets, the signal of the model at time t is given

[1]CoRoT: **Co**nvection **Ro**tation and planetary **T**ransits; European space telescope

by

$$\phi(t) = f^* - \sum_{j=1}^{M} \chi_j^t d_j, \tag{1}$$

where f^* denotes a further parameter describing the regular flux (luminosity) of the host star, and χ_j^t indicates if planet j is transiting at time t and is given by

$$\chi_j^t = \begin{cases} 1 & \text{if } \tau_j < t \bmod p_j \leq \tau_j + l_j \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Parameter M needs to be specified, to avoid the danger of overfitting when including it into the optimization process. The observed data series is given by a list $\{(t_i, f_i)\}, 1 \leq i \leq N$, where t_i denotes a particular observation time and f_i the observed photon flux (i.e. luminosity) at that given time. Let further

$$m_j = (p_j, l_j, d_j, \tau_j) \tag{3}$$

and hence $\vec{m}$ be the vector of all model parameters (except f^*). The overall quality of the fit can be characterized by the root mean square error

$$f(\vec{m}, f^*, \vec{t}, \vec{f}) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (f_i - \phi(t_i))^2}. \tag{4}$$

The objective is to find a parameter setup for $\vec{m}$ and f^* minimizing Eq. (4), i.e. to find a model with minimal deviation from the observations.

Due to stellar fluctuations and measurement errors real-world instances contain noisy signals. The signal-to-noise ratios can be expected to be very low, i.e. the respective values of d_j will be close to the order of magnitude as the standard deviation σ_f of the input values.

3. PREVIOUS WORK

The development of efficient transit detection algorithms has recently gained more interest in the scientific community, as space-based missions like CoRoT [5] or Kepler [17] provide a great amount of observational data. For instance, one observation cycle of CoRoT consists of approximately 10.000 photometric measurements with a resolution of 512 seconds of roughly 12.000 stars. Selected parts can be observed with shorter time windows: the *"alarm mode"* provides a higher resolution of 32 seconds. Other missions like Kepler or the prospected *PLATO* mission [8] will provide even larger data sets. Filtering out a reasonable small subset of candidate planets for a further detailed analysis (including follow-up radial velocity measurements) is thus a challenging problem from a human and computational point of view. The problem is relatively well solved for single planet signals around single host stars but gets much more challenging for planets around binary stars and multiple transiting planets, respectively.

Most of the existing approaches are particularly tailored to the detection of single transit signals. Many algorithms are based on *phase dispersion minimization* [25] where the objective is to find the period in a discretized parameter space that minimizes the scattering of the data folded w.r.t. that period. One of the most popular approaches is the *box fitting least-square algorithm* (BLS) [18]. The search is performed for all periods of a discretized parameter space, the data is then folded according to each of these periods and a box shape is then fitted to the data. A comparison of BLS with a matched-filter approach (see [16] and [7]) is performed in [26]. In [12] the authors combine BLS performed on a coarse grid with a subsequent Newton-Raphson refinement. Further comparisons and analyses are presented in [2]. In [21] the authors propose a wavelet-based algorithm, a heuristic Markov Chain Monte Carlo method has been suggested in [13], and a further alternative method "BAST" can be found in [22]. In the latter approach a box-search for transit-like events is carried out in the unfolded data.

So far, not much effort has been taken in the direction of the development of efficient algorithms for the detection of multi-planet transiting systems (MTSs). An iterative approach, which purely sequentially determines planet fits, is described in [15]. Obviously, such iterative techniques may particularly fail when multiple signals of the same order of magnitude are involved. The major problem is when fitting the signal of one individual planet (probably the one with strongest signals) of a MTS does not allow for the identification of a clear transit-shape due to the interference of the remaining planets signals.

As the majority of existing algorithms operate on binned phase-folded data the remaining signal may be completely vanished as it will be blurred over the phase of the first planet. These effects create the need for a method to simultaneously fitting multiple signals.

So far, no system with multiple transiting planets has been detected – up to now, such systems have only been revealed by the radial velocity method. The current goal is to find multiple systems in the available photometric data. Existing methods are likely to miss such systems in particular, as they would manifest with multiple signals of approximately the same order of magnitude. We therefore deduce the necessity of a fully-automatic procedure being capable of fitting multi-planet transit models to the observation data.

Various applications for evolutionary algorithms in the field of astronomy are outlined in [9] and have since then been successfully applied for many purposes. Fitting procedures based on Evolution Strategies have been considered in [10], where parameters of Keplerian models are successfully determined by fitting them to radial velocity data. In this work we pursue the application of Evolution Strategies to the problem on photometric data as introduced in Section 2. Properties of the search space have been analyzed in [11], coming to the conclusion, that the search landscape does not contain much guiding information, as most parameter configurations do not yield lower values of Eq. 4 than setting $\phi(t)$ to the average value of $\vec{f}$ for all t. If the small basin of attraction w.r.t. the parameters of one particular planet has been found, it is, however, relatively easy to find the corresponding optimum even without application of specialized numeric local optimization methods.

4. IMPROVEMENT AND EVALUATION OF CANDIDATE SOLUTIONS

The overall search process becomes more efficient, when optimal values of depths d_j are automatically derived from p_j, l_j, and τ_j for each planet j. For this purpose we introduce binary flags $(b_1, \ldots, b_M) \in \{0,1\}^M$ for each observation point $o_i = (t_i, f_i), i = 1, \ldots, N$, indicating which planet is transiting at the given time according to the current (partial) model p_j, l_j, τ_j for each $j = 1, \ldots, M$. These flags can be interpreted as integer number with binary representation $b_1 b_2 \ldots b_M \in [0, 2^M - 1]$, implying a partitioning of the set $O = \{o_1, \ldots, o_N\}$ of all observation points $O = O_0 \cup O_1 \cup \ldots \cup O_{2^M - 1}$. Assuming, for example, two planets $M = 2$ we obtain the set of out-of-transit observations O_0, the sets O_1, O_2 of transit events of planets one and two respectively, and the set O_3 where planets one and two are transiting simultaneously.

Optimal transit depths can be derived by minimizing

$$f(\vec{d}) = \sum_{i=1}^{N} \left(f_i - \left(f^* - \sum_{j=1}^{M} \chi_j^i d_j \right) \right)^2, \tag{5}$$

which can be achieved by solving the system of linear equations resulting from

$$\frac{\partial f(\vec{d})}{\partial d_k} = 2 \sum_{i=1}^{N} \left(f_i - f^* + \sum_{j=1}^{M} \chi_j^i d_j \right) \cdot \chi_k^i = 0 \tag{6}$$

for all $k = 1, \dots, M$. Let

$$\hat{f}^K = \sum_{i \in \bigcup_{k \in K} O_k} f_i, \quad K \subseteq \{0, \dots, 2^M - 1\} \tag{7}$$

denote the sum of the observed photon fluxes from groups $\bigcup_{k\in K} O_k$, and $\hat{f} = \sum_{i=1}^N f_i$. Let further $n_K = |\bigcup_{k\in K} O_k|$ and $\tilde{\chi}_j^i, j = 1, \dots, 2^M - 1,\ i = 1, \dots, N$, indicate if observation i belongs to group j. For the case $M = 2$ direct expressions can be derived (see [11]), whereas the general case ($M > 2$) requires the solution of the system of linear equations given by (6). For this purpose we need to rewrite (6) in order to resolve the values of the coefficients of d_k. Let $\Gamma(.)$ denote the set of group indices belonging to the planet indices given as argument(s), i.e. the set of all $l = 1, \dots, 2^M - 1$ where the bitwise logic comparison with all of its arguments yields a positive value. We obtain

$$\sum_{j=1}^{M} n_{\Gamma(j,k)} \cdot d_j = n_{\Gamma(k)} \cdot f^* - \hat{f}^{\Gamma(k)} \tag{8}$$

for each $k = 1, \dots, M$. To give an example, for $k = 2$ we obtain $|O_3 \cup O_7| \cdot d_1 + |O_2 \cup O_3 \cup O_6 \cup O_7| \cdot d_2 + |O_6 \cup O_7| \cdot d_3$ for the term on the left hand side of (8).

4.1 Speeding up the objective function evaluation

By splitting the sum over all observation points in one sum for out-of-transit measurements and another one for in-transit events, we can rewrite (4) as

$$f(\vec{m}, \vec{t}, \vec{f}) = \sqrt{\frac{1}{N} \sum_{i \in O_0} (f_i - f^*)^2 + \frac{1}{N} \sum_{i \in O \setminus O_0} (f_i - \phi(t_i))^2}. \tag{9}$$

We can speed up the objective function evaluation by not iterating over all N observation points when evaluating (4), but only iterating over points where a transit event occurs. To do so we rewrite the first term in the square root of (9) to

$$\frac{1}{N} \left(\sum_{i \in O_0} f_i^2 + n_0 f^{*2} - 2 f^* \sum_{i \in O_0} f_i \right). \tag{10}$$

These three terms can now be reformulated such that no points from O_0 need to be considered for evaluating (4), when $\hat{f}_{\text{sq}} = \sum_{i=1}^N f_i^2$ and $\hat{f}$ are computed in advance. In particular we get

$$\sum_{i \in O_0} f_i^2 = \hat{f}_{\text{sq}} - \sum_{i \in O \setminus O_0} f_i^2, \tag{11a}$$

$$f^{*2} = (\hat{f} - \sum_{i \in O \setminus O_0} f_i), \tag{11b}$$

$$\sum_{i \in O_0} f_i = \hat{f} - \sum_{i \in O \setminus O_0} f_i, \tag{11c}$$

where all terms from the right hand sides can either be computed in advance or contain in-transit points exclusively.

However, given the model parameters $\vec{m}$ we do not know their implied index sets O_k in advance. In order to avoid iterating over the whole index set we use the following algorithm for a fast computation of (4), which uses estimations of the start-indices of the next transit to "jump" over the out-of-transit parts. For this we employ an array S containing pairs (idx, t) of next transit start indices and times for each planet at each time. The final outcome is a list T of index-pairs containing the in-transit intervals.

Algorithm 1: build-transit-indicator()

```
1  build initial start-list $S$
2  $i \leftarrow min_{idx}\ S$
3  while $(i < N)$ do
4      $i_{\text{start}} \leftarrow i$
5      while $i < N \wedge \tilde{\chi}^i > 0$ do  $i \leftarrow i + 1$
6      $i_{\text{end}} \leftarrow i$
7      $T \leftarrow T \cup (i_{\text{start}}, i_{\text{end}})$
8      if $i < N$ then
9          update-start-list($i$)
10         $i \leftarrow (\text{argmin}_t\ S).\text{idx}$
11     end
12 end
```

Algorithm 1 shows the main procedure of building up T. In a first step, start-list S is initialized (line 1) by determining the first transit event of each planet. In the subsequent loop (line 3) index-pairs of transit events are successively added by extracting the minimum index from S (line 9), incrementing the index as long as staying in-transit (line 5), and performing respective updates of the start-list S. Function $\tilde{\chi}^i = \sum_{j=1}^{2^M-1} j \cdot \tilde{\chi}_j^i$ indicates which planets are transiting at time t_i, i.e. $\tilde{\chi}^i = 0$ if there are no transits at t_i, and for instance $\tilde{\chi}^i = 2^{(p-1)} + 2^{(q-1)}$ if planets p and q are transiting simultaneously at time t_i. Hence $\tilde{\chi}^i$ is the interpretation of the binary flags $b_1 b_2 \dots b_M$ introduced at the beginning of this section as integer numbers.

Algorithm 2: update-start-list($i_{t_{\text{curr}}}$)

```
1  for $j = 1, \dots, M$ do
2      $i_{\text{new}} \leftarrow i_{t_{\text{curr}}}$
3      while $i_{\text{new}} \leq i_{t_{\text{curr}}}$ do
4          $t \leftarrow$ find-next-transit-start-time($j$)
5          $i_{\text{new}} \leftarrow$ find-time-index($i_{t_{\text{curr}}}, t$)
6      end
7      $S[j].\text{idx} \leftarrow i_{\text{new}}$
8      $S[j].t \leftarrow t$
9  end
```

Algorithm 2 shows the update procedure of start-list S. The argument $i_{t_{\text{curr}}}$ denotes the current position in the time array, i.e. we are looking for the next transit-start-times after $t_{i_{t_{\text{curr}}}}$. This is achieved by iteratively searching for the next transit times of each planet (w.r.t. the current status of S) as long as it remains smaller than $t_{i_{t_{\text{curr}}}}$ (see lines 3-6 in Algorithm 2). For this purpose function `find-next-transit-start-time`(t) (Algorithm 3) is used to determine the time t on the one hand (line 4), and on the other hand function `find-time-index`$(i_{t_{\text{curr}}})$ (Algorithm 4) to find the corresponding index (line 5).

Function `find-next-transit-start-time`(j) (Algorithm 3) returns the next transit start time for planet j w.r.t. the current status of S. If S has already been initialized (line 1) we simply have to add the period p_j to the current value. Otherwise the first transit start-time according to t_0 has to be determined. In line 7 of Algorithm 3 we assign the relative position in phase of observation time t_0 to variable t, taking care of the special case $\tau_j + l_j > p_j$, which is necessary for the actual calculation of the transit start times in line 8.

Algorithm 3: find-next-transit-start-time(j)

```
1  if $S[j].t$ not uninitialized then
2      // this is not the first entry
3      // $\rightarrow$ simply add period
4      return $t + p_j$
5  else
6      // first transit
7      $t \leftarrow \begin{cases} t_0 \bmod p_j + p_j & \text{if } \tau_j + l_j > p_j \wedge t < \tau_j \\ t_0 \bmod p_j & \text{otherwise} \end{cases}$
8      $t^+ \leftarrow \begin{cases} t_0 + (\tau_j - t) & \text{if } t < \tau_j + l_j \\ t_0 + (\tau_j + p_j - t) & \text{otherwise} \end{cases}$
9      return $t^+$
10 end
```

Given these transit start times, the corresponding indices of $\vec{t}$ still need to be determined. Although the input-data is mostly equally sampled, there may be missing points or even sequences. Thus we use function `find-time-index` (Algorithm 4) to search for the correct start-index. This procedure is based on iterative predictions of the position in $\vec{t}$ (line 6 and 7) by dividing the interval from the current time to the search-time by Δt_{avg}, the average time interval $(t_{i+1} - t_i)$.

Algorithm 4: find-time-index($i_{t_{\text{curr}}}, t$)

```
1  $t_{\text{curr}} \leftarrow t_{i_{t_{\text{curr}}}}$
2  $\Delta t \leftarrow t - t_{\text{curr}}$
3  $i_{\text{est}} \leftarrow i_{t_{\text{curr}}} + \Delta t / \Delta t_{\text{avg}}$
4  while $\neg(t_{i_{\text{est}}} \leq t \wedge t_{i_{\text{est}}+1} > t)$ do
5      $t_{\text{curr}} \leftarrow t_{i_{\text{est}}}$
6      $\Delta t \leftarrow t - t_{\text{curr}}$
7      $i_{\text{est}} \leftarrow i_{\text{est}} + \Delta t / \Delta t_{\text{avg}}$
8  end
9  return $i_{\text{est}}$
```

Note that the calculation of transit depths according to (6) requires an additional iteration over all in-transit data-points, whereas using it together with the transit-indicator does not imply a further additional iteration, as the information for calculating the transit depths is already gathered by building up the transit-indicator T.

4.2 Parallelization by Utilization of the Graphics Processing Unit (GPU)

Population based metaheuristics typically provide the possibility for easy parallelization, as operations like variation or evaluation usually have to be performed for many candidate solution. In our case there is an additional potential for parallelization, as for each fitness function evaluation a huge amount of data points needs to be compared to the model, comprising identical operations for each such point. This aspect has already been successfully exploited in other contexts, see e.g. [23].

Such situations are usually classified as *Single Instruction, Multiple Data* (SIMD). Over the recent years, the deployment of *graphics processing units* (GPUs) became very popular for general computing purposes possibly subject to parallelization, as they are especially designed for high parallelism. Due to open application programming interfaces it is relatively comfortable to use the streaming multiprocessors (SMs) of a graphics card for *general purpose computing on GPUs* (GPGPU); see for instance [20] for a broad overview. For our implementation we specifically used the NVIDIA CUDA interface [1]. If parallelization is performed on the level of whole candidate solutions *(coarse-grained approach)*, we usually do not fully load the GPU. As a consequence the gain is only moderate. In order to exploit the capabilities of the GPU in a better way, an additional parallelization on data-level is more fruitful (*fine-grained approach*).

As first step when evaluating a candidate model, we need to calculate the optimal transit depths for each planet. In order to solve Eqs. (8) we need to iterate over all data points to compute $\hat{f}^K$. These operations can be carried out in parallel for equally sized groups of data, merging the results afterwards. Having calculated $d_j, j = 1, \ldots, M$, we need to compute the result of Eq. (4), again requiring to iterate over all data points. The algorithms described in Section 4.1 by-pass the iterations over the whole index set for the execution on CPUs, which can however not be transfered to the GPU, as the resulting code provides no direct way for parallelization. So the same splitting and merging steps as done for the calculation of d_j are carried out again. To fully benefit from the GPU's computational power it needs to be carefully decided which entities (blocks of data, groups of candidate solutions) of which size are assigned to the SMs. This issue is addressed in more detail in Section 6.1, where corresponding results are also presented.

5. GLOBAL OPTIMIZATION BY EVOLUTION STRATEGIES

To solve the global parameter optimization problem, we use an *Evolution Strategy* (ES) [3], as this particular metaheuristic turned out to be adequate for this problem in a preceding comparison [11]. The ES can be classified as a (μ, λ)-ES with self-adaptation of strategy parameters [24], where μ denotes the size of the population and λ the number of offsprings created in each iteration. For each parameter x_k of each individual in the population, mutation is performed in the following, usual way,

$$x'_k = x_k + N_k(0, \sigma'_k), \tag{12}$$

where $N(0, \sigma)$ denotes the Gaussian distribution having a mean value of zero and a standard deviation σ. The values $\sigma'_k \in [10^{-5}, \infty]$ are used as strategy parameters which themselves undergo the process of mutation, given by

$$\sigma'_k = \sigma_k \cdot e^{N(0,\tau_0) + N_k(0,\tau)}. \tag{13}$$

This self-adaptive process attempts to exploit properties of the fitness landscape and thus facilitates an efficient search process. It turned out to be advantageous to use a variant of elitist-selection which creates the new population by deterministically taking the best μ individuals from the μ parents and λ offsprings, but taking at most $\hat{\mu}$ individuals from the parents. Hence, our selection is in fact in-between $(\mu + \lambda)$-selection and (μ, λ)-selection. The reason behind this approach is to support exploitation of the regions around the best solutions found so far, which is rather important, as the basins of attraction are relatively small. Not exclusively using the $(\mu + \lambda)$-strategy reduces the risk of getting stuck to local optima. Following the suggestions in [24], we set parameters $\tau = \left(\sqrt{2\sqrt{4 \cdot M + 1}}\right)^{-1}$ and $\tau_0 = \left(\sqrt{2(4 \cdot M + 1)}\right)^{-1}$. If some

Table 1: Test-instances (no. 100-111) with corresponding success ratios and average running times for signals with one artificial planet.

Parameters				(% opt.)			$t_{\text{avg}}[s]$		
p	l	d	τ	$\sigma_N : 0$	100	300	0	100	300
1.0	0.10	100.0	0.5	100	98	82	13	20	118
1.0	0.10	500.0	0.5	100	100	100	12	9	18
2.0	0.10	100.0	0.5	100	100	72	6	6	146
2.0	0.10	500.0	0.5	100	100	100	7	6	6
2.0	0.10	100.0	0.5	100	100	100	7	6	6
2.0	0.05	500.0	0.5	100	100	n/a	6	8	n/a
1.0	0.05	500.0	0.5	98	100	80	23	21	81
1.0	0.05	300.0	0.5	96	98	64	33	27	161
1.0	0.05	100.0	0.5	100	94	n/a	22	50	n/a
1.0	0.02	500.0	0.5	80	82	32	82	63	302
1.0	0.02	300.0	0.5	74	46	8	95	154	421
1.0	0.02	100.0	0.5	60	44	n/a	121	257	n/a

depth is set to 0.0 – implying that this particular planet-model does not improve the quality of the fit at all – a new random planet is created on this position, which might increase diversity among the population. Prior to mutation, recombination is applied with probability $P_R \in [0,1]$. We use the intermediate recombination, given by

$$x'_k = \alpha_k \cdot x^1_k + (1-\alpha_k)x^2_k, \tag{14}$$

where x^1_k and x^2_k denote the parameters of the parents and α_k is a uniform random number from the interval $[-\beta, 1+\beta]$ for each parameter k, where $\beta = \frac{1}{2}$ turned out to be most successful.

6. RESULTS

For a comprehensive evaluation of the presented algorithms we created artificial test-instances with up to three planets. Stellar jitter and measurement errors of real data instances are simulated by adding Gaussian random values to each data point in the artificial signal. For each configuration three instances with different noise-levels are created, $\sigma_N = 0, \sigma_N = 100$ and $\sigma_N = 300$.

For our computational experiments we use the following parameter setting that has been determined in preliminary tests: $\mu = 100, \lambda = 500, \hat{\mu} = 50$. Prior to mutation we perform intermediate recombination for the strategy parameters and parameters with a probability of $P_R = 0.8$, which showed the overall best performance within preliminary tests. The maximum number of iterations was set to 2000 and no time limit has been imposed, but the runs have been stopped as soon as the correct solution has been found. All tests have been performed on a cluster consisting of Intel Xeon E5540 processors operating at 2.53 GHz and having 24 GB for totally 8 cores.

Let $\vec{m}' = (\vec{m}, f^*)$, and $f(\vec{m}'_g)$ denote, as shorthand for $f(\vec{m}'_g, \vec{t}, \vec{f})$, the objective function value of the *generated* signal with parameters $\vec{m}'_g$. We call all solutions with $f(\vec{m}') \leq f(\vec{m}'_g) + \varepsilon$ "optimal", for our experiments we used $\varepsilon = 10^{-5}$. It is important to note, that for noisy signals more than one solution $\vec{m}'$ with $f(\vec{m}') \leq f(\vec{m}'_g) + \varepsilon$ will exist, as the noise added to the generated signal will render $\vec{m}'_g$ subobtimal. Our experiments showed that typically all of these solutions closely correspond to the parameters used for signal generation, i.e. no significant deviations have been observed. In practice, finding any of these solutions would be sufficient for the detection of the planetary system.

Tables 1 and 2 show results for test instances containing one and two artificial signals, respectively. We report the percentage of runs

Table 2: Test-instances (no. 210-219) with corresponding success ratios and average running times for signals with two artificial planets.

Parameters				(% opt.)			$t_{\text{avg}}[s]$		
p	l	d	τ	$\sigma_N : 0$	100	300	0	100	300
1.0	0.10	500.0	0.5	88	94	80	251	161	318
2.2	0.10	500.0	1.0						
1.0	0.10	500.0	0.5	100	100	78	92	91	332
2.2	0.10	300.0	1.0						
1.0	0.10	300.0	0.5	56	54	14	532	501	917
2.2	0.10	500.0	1.0						
1.0	0.05	500.0	0.5	86	70	20	227	377	655
2.2	0.05	500.0	1.0						
1.0	0.05	400.0	0.5	70	76	60	312	276	378
7.5	0.20	500.0	1.0						
1.0	0.10	400.0	0.5	64	70	84	487	430	244
7.5	0.50	500.0	1.0						
1.0	0.05	400.0	0.5	28	26	14	615	705	841
3.1	0.10	500.0	1.0						
1.0	0.05	500.0	0.5	64	76	34	410	313	660
3.1	0.10	400.0	1.0						
1.0	0.05	500.0	0.5	90	88	44	228	207	608
3.1	0.10	300.0	1.0						
1.0	0.05	500.0	0.5	94	88	56	215	244	525
3.1	0.10	200.0	1.0						

where "optimal" solutions have been obtained and average running times for three different noise levels σ_N. Fifty independent runs have been performed for each such case. For some instances with $\sigma = 300$ no results are available (indicated by "n/a"), as the algorithm stopped prematurely because of solutions having lower fitness values than the intended optimal solution corresponding to the original artificial signal. We can observe, that the algorithm is able to find the original signal with high probability, in particular for the case of one planet with a probability close to one for almost all instances. As expected, the detection efficiency decreases significantly in the case the noise exaggerates the signal power. In the case of two artificial signals, the probability of signal detection is still high enough to provide a satisfactory detection probability when the algorithm is executed two or three times on the particular data instance.

Table 3 shows results for three-planet signals, average computational times for a full run are roughly 2000 seconds. What can already be observed from the one- and two-planet results is (the obvious) property that detection efficiency is highly dependent on the signal power, which is given by the parameters d and l. To study this effect in more detail, we systematically vary the parameters d and l and keep the other parameters fixed to the values $p_1 = 1.0, p_2 = 3.3, p_3 = 7.7, \tau_1 = 0.5, \tau_2 = 1.0, \tau_3 = 2.5$. A systematic study of the detection efficiency in dependence of the other parameters is beyond the scope of this work. However, preliminary results indicate that their variation has less impact than the variation of d and l. It can be observed that signals containing more noise are often detected more frequently, which is due to the particular stopping criterion and our definition of optimal solutions mentioned in the last paragraph. Most notably, the results show that even a relatively large amount of *white* noise in general does not have a substantial negative impact on detection efficiency.

All results have been computed by using the parameter-space reduction and advanced fitness-function evaluation method as described in Section 4. The optimal calculation of the depths significantly improves the ability of the algorithm to improve existing solutions fast. The advanced fitness function evaluation procedure improves the overall running times of up to a factor of four.

The results indicate that the algorithm is a promising tool for multi-planet transit detection, as, at least for the considered kinds of instances, optimal solutions can be obtained with relatively high probability and reasonable running times.

6.1 GPU Results

In this section we present results and hardware specific parameter settings regarding our GPU-implementation of the ESTEX algorithm. As test platform we used an AMD Athlon 64 3200+ PC with 2000 MHz with a *GeForce 9600 GT* GPU with eight streaming multiprocessors (SMs). Each of these SMs can execute 32 threads, called *warps*. It is usually a good practice to pass larger *blocks* (groups of threads) to the GPU as particular threads of a warp may be stalled due to relatively time consuming instructions, such as memory access instructions [1]. In our case, the number of SM registers is the limiting factor. There are 8192 registers available on each SM, but each thread is assigned its own physical registers, which then cannot be used by other threads until the considered block is finished. As each of these threads require 23 registers to perform its instructions, we obtain an upper bound of $8192/23 = 356$ for the number of threads to be assigned to one SM. It is however only possible to specify numbers of threads corresponding to powers of two, so we obtain $8192/32 = 256$ as the number of threads assigned to one SM. From this we can see, that within the coarse grained approach we do not fully load the GPU, even if the number of individuals corresponds to the number of threads that can be executed in parallel on the GPU. This expected behaviour is also confirmed by the experiments summarized in Table 4.

We therefore group the input-data (of usually 10^4 points) in *data-blocks* and evaluate 256 candidate models on one SM. Using a data-block size of 1250 therefore fully loads the eight available SMs. Table 4 gives an overview of the running-times achieved by various data-block sizes for a typical two-planet instance. The best considered assignment to the eight available SMs yields a speedup factor of almost forty, and we can expect this factor to grow almost linearly with the number of available SMs. For three-planet systems the situation gets more complicated, as we are required to solve a system of linear equations (8). Nevertheless, we are confident to be able to nearly conserve the speedup factor, by using GPU-implementations of solvers for systems of linear equations like [6].

Compared to the CPU variant using the improved objective function evaluation methods from Section 4.1 (which probably cannot be reasonably implemented on GPUs) we still have a speedup of a factor more than eight. Hence the GPU approach makes particularly sense for the application on modern workstations for application scenarios when scientists want to compute fits for specifically selected instances. On recently emerging GPU clusters this approach is also auspicious for the processing of large-scale data sets.

7. REAL DATA INSTANCES

Although many single planet fitting-algorithms use box-shaped models, more accurate models significantly improve the algorithms ability to find multiple planetary systems. As real transit signatures deviate from the simple box shape, residuals resulting from subtracting an (optimal) box shape remain in the signal and will due to their periodicity likely obscure further planetary signals. This is particularly true for heuristic search processes as such artifacts might introduce strong local optima possibly attracting many solution candidates and therefore leading to convergence to a suboptimal solution even in the case these artifacts have less power than the real further planetary signals. We are hence interested in a very accurate description of the real transit shape. A reasonable accurate transit model is determined by the relative size of the planet to the star as well as the *impact parameter*, which is the distance of the transiting line to the center of the host star, and the stellar limb darkening. Exact transit models have been developed in [19] and can be easily integrated into the presented algorithms by the instruction of further model parameters. For preliminary experiments with real photometric data we applied the following simpler transit model by introducing one further parameter $s_j \in [0,1], 1 \le j \le M$. We therefore redefine $m_j = (p_j, l_j, d_j, \tau_j, s_j)$. The following sigmoid-shaped functions can be used to obtain a more accurate description of the transit shape:

$$u_1(t) = \left(1 + e^{\xi - 2\cdot\xi\cdot t}\right)^{-1}, \tag{15}$$

$$u_2(t) = \left(1 + e^{-\xi + 2\cdot\xi\cdot t}\right)^{-1}. \tag{16}$$

Within these formulas the constant ξ used as a factor for t in the exponent determines the slope of the sigmoid shape, the additive term ξ transforms the sigmoid shape in interval $[0,1]$. Reasonable values are $5 \le \xi \le 10$. With $u_1(t)$ and $u_2(t)$ we can use the following functions to describe the ingress and egress phase of the transit.

$$\tilde{u}_1(t, m_j) = u_1\left(\frac{2(t \bmod p_j - \tau_j)}{s_j \cdot l_j}\right), \tag{17}$$

$$\tilde{u}_2(t, m_j) = u_2\left(\frac{2(t \bmod p_j - \tau_j - l_j + \frac{s_j}{2} l_j)}{s_j \cdot l_j}\right). \tag{18}$$

The whole transit-shape is then given by

$$\chi_j(s,t) = \begin{cases} \tilde{u}_1(t, m_j) & \text{if } \tau_j < t \bmod p_j < \tau_j + \frac{s_j}{2} l_j \\ \tilde{u}_2(t, m_j) & \text{if } \tau_j + l_j - \frac{s_j}{2} l_j < t \bmod p_j < \tau_j + l_j \\ 1 & \text{if } \tau_j + \frac{s_j}{2} l_j < t \bmod p_j \le \tau_j + l_j - \frac{s_j}{2} l_j \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

instead of Eq. (2) used for the box-model, and now provides continuous values indicating the percentage of total transit depth for planet j at time t.

Small values of s approximate a V-shaped transit, whereas intermediate values correspond to a typical (planetary) U-shaped signal.

Table 4: Running times of instance art-215-j when using GPU with various parameter settings compared to CPU version (with and without transit indicator). Parameter settings: $\mu = 50, \lambda = 230, it_{\max} = 500$.

Processing Unit	Ind. per SM	Data-blocks	t[s]
1 SM	256	coarse	77
2 SM	128	coarse	72
4 SM	64	coarse	72
8 SM	32	coarse	70
2 SM	256	5000	40
4 SM	256	2500	22
8 SM	256	1250	14
CPU (transit indicator)	n/a	n/a	119
CPU (no transit indicator)	n/a	n/a	528

Table 3: Success ratios for three-planet artificial test instances (no. 300-363). All of these instances have in common the parameters $p_1 = 1.0, p_2 = 3.3, p_3 = 7.7, \tau_1 = 0.5, \tau_2 = 1.0, \tau_3 = 2.5$.

Parameters		(% opt.)			Parameters		(% opt.)			Parameters		(% opt.)		
l_1, l_2, l_3	d_1, d_2, d_3	$\sigma_N : 0$	100	300	l_1, l_2, l_3	d_1, d_2, d_3	$\sigma_N : 0$	100	300	l_1, l_2, l_3	d_1, d_2, d_3	$\sigma_N : 0$	100	300
0.05, 0.15, 0.40	400, 400, 400	30	14	46	0.10, 0.15, 0.80	400, 200, 400	56	40	64	0.10, 0.30, 0.40	400, 200, 200	40	30	26
0.10, 0.15, 0.40	400, 400, 400	28	30	74	0.10, 0.30, 0.80	400, 200, 400	30	28	70	0.05, 0.30, 0.80	400, 200, 200	8	20	20
0.05, 0.30, 0.40	400, 400, 400	36	34	56	0.05, 0.15, 0.40	400, 400, 200	22	56	50	0.10, 0.15, 0.80	400, 200, 200	22	42	28
0.05, 0.15, 0.80	400, 400, 400	26	30	24	0.10, 0.15, 0.40	400, 400, 200	46	52	92	0.10, 0.30, 0.80	400, 200, 200	20	8	22
0.10, 0.30, 0.40	400, 400, 400	44	34	68	0.05, 0.30, 0.40	400, 400, 200	28	58	58	0.05, 0.15, 0.40	200, 400, 200	40	30	2
0.05, 0.30, 0.80	400, 400, 400	24	26	38	0.05, 0.15, 0.80	400, 400, 200	14	24	8	0.10, 0.15, 0.40	200, 400, 200	40	64	70
0.10, 0.15, 0.80	400, 400, 400	32	36	74	0.10, 0.30, 0.40	400, 400, 200	36	48	82	0.05, 0.30, 0.40	200, 400, 200	36	44	36
0.10, 0.30, 0.80	400, 400, 400	44	26	70	0.05, 0.30, 0.80	400, 400, 200	10	40	58	0.05, 0.15, 0.80	200, 400, 200	24	8	22
0.05, 0.15, 0.40	200, 400, 400	24	28	28	0.10, 0.15, 0.80	400, 400, 200	6	16	22	0.10, 0.30, 0.40	200, 400, 200	54	64	68
0.10, 0.15, 0.40	200, 400, 400	38	36	58	0.10, 0.30, 0.80	400, 400, 200	30	26	84	0.05, 0.30, 0.80	200, 400, 200	24	44	42
0.05, 0.30, 0.40	200, 400, 400	48	50	16	0.05, 0.15, 0.40	200, 200, 400	44	36	18	0.10, 0.15, 0.80	200, 400, 200	24	36	56
0.05, 0.15, 0.80	200, 400, 400	36	32	28	0.10, 0.15, 0.40	200, 200, 400	46	56	46	0.10, 0.30, 0.80	200, 400, 200	50	80	66
0.10, 0.30, 0.40	200, 400, 400	56	68	56	0.05, 0.30, 0.40	200, 200, 400	36	30	44	0.05, 0.15, 0.40	200, 200, 200	38	46	36
0.05, 0.30, 0.80	200, 400, 400	36	38	28	0.05, 0.15, 0.80	200, 200, 400	40	22	20	0.10, 0.15, 0.40	200, 200, 200	68	68	50
0.10, 0.15, 0.80	200, 400, 400	46	38	66	0.10, 0.30, 0.40	200, 200, 400	40	30	62	0.05, 0.30, 0.40	200, 200, 200	44	52	14
0.10, 0.30, 0.80	200, 400, 400	50	44	30	0.05, 0.30, 0.80	200, 200, 400	26	42	38	0.05, 0.15, 0.80	200, 200, 200	28	36	30
0.05, 0.15, 0.40	400, 200, 400	28	22	60	0.10, 0.15, 0.80	200, 200, 400	46	52	74	0.10, 0.30, 0.40	200, 200, 200	44	52	64
0.10, 0.15, 0.40	400, 200, 400	52	40	60	0.10, 0.30, 0.80	200, 200, 400	46	50	85	0.05, 0.30, 0.80	200, 200, 200	16	44	28
0.05, 0.30, 0.40	400, 200, 400	26	34	66	0.05, 0.15, 0.40	400, 200, 200	16	18	54	0.10, 0.15, 0.80	200, 200, 200	32	46	68
0.05, 0.15, 0.80	400, 200, 400	24	20	62	0.10, 0.15, 0.40	400, 200, 200	44	20	76	0.10, 0.15, 0.80	200, 200, 200	44	34	72
0.10, 0.30, 0.40	400, 200, 400	40	32	50	0.05, 0.30, 0.40	400, 200, 200	26	28	68					
0.05, 0.30, 0.80	400, 200, 400	26	24	54	0.05, 0.15, 0.80	400, 200, 200	4	4	22					

Values close to zero correspond to a box like shape, which is typical for relatively small planets producing central transits.

In order to determine the optimal transit depths we do, however, still implicitly use the rectangular shape, as it enables to take average values over the particular index sets $O_i,\ i \in [0, 2^M]$, as a basis for the calculations given by Eqs. (8). However, by this procedure we systematically underestimate the transit depths in roughly linear dependence of parameter s_j. We therefore approximately correct the depths according to the following formula

$$d'_j = d_j \cdot (1 + s_j), \tag{20}$$

which turned out to work well by computational experiments.

First experiments with real photometric data series already showed that this transit model already overcomes the discussed issues of the box-shape and therefore significantly improves convergence properties.

A further important issue regarding the application to real photometric data is how to identify promising planetary candidates amongst the huge set of computed fits. Having computed fits for particular data instances we need to discriminate between just having adjusted the model parameters to noise and artifacts, or having found plausible transit signals. The primary parameter for such discriminative test is the signal-to-noise ratio in the particular transit-phases compared to the signal power (i.e. transit length and depth) of the model, e.g. see [16, 18]. As a first simple approach we can for each planet consider the ratio of the standard deviations within each transit phase of the residual signal, i.e. the obtained fit subtracted from the data, to the corresponding transit depth. For a plausible fit the depth must be significantly larger than the order of magnitude of scattering within the transit phase. Having found a plausible fit, there still is the chance of just facing stellar variability or eclipsing binary stars. Many of these cases can likely be eliminated by checking the parameters for plausibility, see [27]. The development of more sophisticated and reliable methods to identify promising candidates, however, remains future work, as this work primarily deals with optimization aspect of the problem.

8. CONCLUSIONS AND FUTURE WORK

The computational experiments presented in Section 6 clearly show that the ESTEX algorithm performs quite well on the considered test instances. In the course of our computational experiments the methods to speed up the objective function evaluation in Section 4.1 turned out to reduce the overall computation time of up to a factor of four. This method, as well as the parameter space reduction by the calculation of optimal depths according to some given partial solution is not only applicable to the ESTEX algorithm, but also to any other metaheuristic global optimization procedure. A further improvement can be obtained by parallelizing the computations by using graphics processing units. A speedup of a factor up to forty compared to the naive implementation, and more than eight compared to the algorithm using the improved evaluation method can be already achieved on current medium-class GPUs.

In order to process large data sets as provided by space telescopes like CoRoT or Kepler, a single computer will not be able to perform the computations within a reasonable amount of time like days or weeks. This is however no limitation, as (CPU) computing clusters are available at many academic institutions. The most appropriate computing environment would though be a GPU-grid supporting a highly parallelized execution of the algorithms. However, such GPU-grids are currently not as prevalent as CPU-grids, justifying the CPU-variant of the ESTEX algorithm. Furthermore it is currently the only way to handle models of more than two planets.

To our best knowledge, the presented algorithms are the first fully-automatic methods for fitting multi-planet transit models to photometric time-data series. Hence these algorithms might be a useful tool for the discovery of the first planetary system with multiple transiting planets.

9. REFERENCES

[1] CUDA Programming Guide 2.3. http://developer.download.nvidia.com/compute/cuda/2_3/toolkit/docs/NVIDIA_CUDA_Programming_Guide_2.3.pdf, 2009.

[2] S. Aigrain and M. Irwin. Practical planet prospecting. *Monthly Notices of the Royal Astronomical Society*, 350:331–345, May 2004.

[3] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.

[4] A. Baglin. COROT: A minisat for pionnier science, asteroseismology and planets finding. *Advances in Space Research*, 31:345–349, 2003.

[5] P. Barge, A. Baglin, M. Auvergne, J.-T. Buey, C. Catala, E. Michel, W. W. Weiss, M. Deleuil, L. Jorda, C. Moutou, and COROT Team. CoRoT: a first space mission to find terrestrial planets. In F. Casoli, T. Contini, J. M. Hameury, and L. Pagani, editors, *SF2A-2005: Semaine de l'Astrophysique Francaise*, pages 193–199, Dec. 2005.

[6] S. Barrachina, M. Castillo, F. D. Igual, R. Mayo, and E. S. Quintana-Ortí. Solving dense linear systems on graphics processors. In *Euro-Par '08: Proceedings of the 14th international Euro-Par conference on Parallel Processing*, pages 739–748, Berlin, Heidelberg, 2008. Springer-Verlag.

[7] P. Bordé, F. Fressin, M. Ollivier, A. Léger, and D. Rouan. Transdet: a Matched-filter based Algorithm for Transit Detection – Application to Simulated COROT Light Curves. In C. Afonso, D. Weldrake, & T. Henning, editor, *Transiting Extrapolar Planets Workshop*, volume 366 of *Astronomical Society of the Pacific Conference Series*, pages 145–151, July 2007.

[8] C. Catala. PLATO: PLAnetary Transits and Oscillations of stars. *Experimental Astronomy*, 23:329–356, Mar. 2009.

[9] P. Charbonneau. Genetic Algorithms in Astronomy and Astrophysics. *Astrophysical Journal Supplement*, 101:309–334, Dec. 1995.

[10] A. M. Chwatal and G. R. Raidl. Determining orbital elements of extrasolar planets by evolution strategies. In R. M.-D. et al., editor, *Computer Aided Systems Theory – EUROCAST 2007*, volume LNCS 4739 of *LNCS*, pages 870–877, 2007. Eleventh International Conference on Computer Aided Systems Theory.

[11] A. M. Chwatal and G. R. Raidl. Fitting rectangular signals to time series data by metaheuristic algorithms. In R. Moreno-Díaz et al., editor, *Computer Aided Systems Theory – EUROCAST 2009*, volume LNCS 5717 of *LNCS*, pages 649–656, 2009. Twelfth International Conference on Computer Aided Systems Theory.

[12] A. Collier Cameron, D. Pollacco, R. A. Street, T. A. Lister, R. G. West, D. M. Wilson, F. Pont, D. J. Christian, W. I. Clarkson, B. Enoch, A. Evans, A. Fitzsimmons, C. A. Haswell, C. Hellier, S. T. Hodgkin, K. Horne, J. Irwin, S. R. Kane, F. P. Keenan, A. J. Norton, N. R. Parley, J. Osborne, R. Ryans, I. Skillen, and P. J. Wheatley. A fast hybrid algorithm for exoplanetary transit searches. *Monthly Notices of the Royal Astronomical Society*, 373:799–810, Dec. 2006.

[13] A. Collier Cameron, D. M. Wilson, R. G. West, L. Hebb, X. Wang, S. Aigrain, F. Bouchy, D. J. Christian, W. I. Clarkson, B. Enoch, M. Esposito, E. Guenther, C. A. Haswell, G. Hébrard, C. Hellier, K. Horne, J. Irwin, S. R. Kane, B. Loeillet, T. A. Lister, P. Maxted, M. Mayor, C. Moutou, N. Parley, D. Pollacco, F. Pont, D. Queloz, R. Ryans, I. Skillen, R. A. Street, S. Udry, and P. J. Wheatley. Efficient identification of exoplanetary transit candidates from SuperWASP light curves. *Monthly Notices of the Royal Astronomical Society*, 380:1230–1244, Sept. 2007.

[14] H. Deeg, J. A. Belmonte, and A. Aparicio, editors. *Extrasolar Planets*. Cambridge University Press, 2008.

[15] J. M. Jenkins, H. Chandrasekaran, D. A. Caldwell, C. Allen, S. T. Bryson, N. M. Batalha, and W. J. Borucki. Detecting Multiple Transiting Planets with the Kepler Mission. volume 38 of *Bulletin of the American Astronomical Society*, page 192ff., May 2007.

[16] J. M. Jenkins, L. R. Doyle, and D. K. Cullers. A Matched Filter Method for Ground-Based Sub-Noise Detection of Terrestrial Extrasolar Planets in Eclipsing Binaries: Application to CM Draconis. *Icarus*, 119:244–260, Feb. 1996.

[17] D. G. Koch, W. Borucki, L. Webster, E. Dunham, J. Jenkins, J. Marriott, and H. J. Reitsema. Kepler: a space mission to detect earth-class exoplanets. volume 3356 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 599–607, Aug. 1998.

[18] G. Kovács, S. Zucker, and T. Mazeh. A box-fitting algorithm in the search for periodic transits. *Astronomy and Astrophysics*, 391:369–377, Aug. 2002.

[19] K. Mandel and E. Agol. Analytic Light Curves for Planetary Transit Searches. *The Astrophysical Journal*, 580:L171–L175, Dec. 2002.

[20] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips. GPU computing. *Proceedings of the IEEE*, 96(5):879–899, May 2008.

[21] C. Régulo, J. M. Almenara, R. Alonso, H. Deeg, and T. Roca Cortés. TRUFAS, a wavelet-based algorithm for the rapid detection of planetary transits. *Astronomy and Astrophysics*, 467:1345–1352, June 2007.

[22] S. Renner, H. Rauer, A. Erikson, P. Hedelt, P. Kabath, R. Titz, and H. Voss. The BAST algorithm for transit detection. *Astronomy and Astrophysics*, 492:617–620, Dec. 2008.

[23] D. Robilliard, V. Marion-Poty, and C. Fonlupt. Population parallel GP on the G80 GPU. In *Genetic Programming*, pages 98–109. 2008.

[24] H.-P. Schwefel. *Numerical Optimization of Computer Models*. Wiley, Chichester, 1981.

[25] R. F. Stellingwerf. Period determination using phase dispersion minimization. *Astrophysical Journal*, 224:953–960, Sept. 1978.

[26] B. Tingley. Improvements to existing transit detection algorithms and their comparison. *Astronomy and Astrophysics*, 408:L5–L7, Sept. 2003.

[27] B. Tingley and P. D. Sackett. A Photometric Diagnostic to Aid in the Identification of Transiting Extrasolar Planets. *Astrophysical Journal*, 627:1011–1018, July 2005.